Numerical Investigation of Parallel-in-Time Methods for Dominantly Hyperbolic Equations

Vom Fachbereich Maschinenbau
an der Technischen Universität Darmstadt
zur Erlangung des akademischen Grades eines
Doktor-Ingenieurs (Dr.-Ing.)
genehmigte

Dissertation

vorgelegt von

Andreas Schmitt, M.Sc.

aus Düsseldorf

Berichterstatter:	Prof. Dr. rer. nat. Michael Schäfer
Mitberichterstatter:	Prof. Dr. rer. nat. Sebastian Schöps
Tag der Einreichung:	23.07.2018
Tag der mündlichen Prüfung:	10.10.2018

Darmstadt 2019

D 17

Schmitt, Andreas
Numerical Investigation of Parallel-in-Time Methods for Dominantly Hyperbolic Equations
Technische Universität Darmstadt
Tag der mündlichen Prüfung: 10.10.2018
Jahr der Veröffentlichung auf TUprints: 2019
URN: urn:nbn:de:tuda-tuprints-83286

Vorwort

Die vorliegende Arbeit "Numerical Investigation of Parallel-in-Time Methods for Dominantly Hyperbolic Equations" ist mein Beitrag zum Forschungsgebiet der zeitparallelen Methoden. Diese Dissertation wurde als Abschlussarbeit meines Promotionsvorhabens an der Technischen Universität Darmstadt am Fachgebiet für Numerische Berechnungsverfahren im Maschinenbau und der Graduiertenschule Computational Engineering verfasst. Mit den Inhalten der Dissertation habe ich mich von Mai 2015 bis Juli 2018 beschäftigt.

Ich möchte mich bei Prof. Michael Schäfer für die Möglichkeit zur Promotion, den Vorschlag des spannenden Themengebietes und die Betreuung bedanken. Insbesondere danke ich Ihm auch für die gegebenen Freiheiten, so dass ich das Thema nach meinen Interessen erforschen konnte. Des Weiteren danke ich Prof. Sebastian Schöps für die Übernahme der Mitberichterstatterschaft und die konstruktiven Gespräche.

Für viele konstruktive Gespräche möchte ich mich auch bei den Mitentwicklern von SWEET bedanken. Ohne den regen fachlichen Austausch mit Martin Schreiber wäre die Arbeit nicht dieselbe. Prof. Pedro Peixoto danke ich unter Anderem für klärende Antworten zu der semi-Lagrangian Methode.

Zudem gilt mein Dank allen Kollegen am FNB und der GSC, die mich über die Jahre begleitet haben, sowohl für den fachlichen Austausch, als auch den Spaß bei verschiedenen Freizeitaktivitäten. Die angenehme Arbeitsumgebung werde ich in Erinnerung behalten. Namentlich erwähnt seien hier Jakob, dessen Bürotüre nie für Disskussionen jeder Art verschlossen war, und Dominic, der in der Phase des Zusammenschreibens ein Leidensgenosse zum gegenseitigen Motivieren war. Christian und Michael bin ich für die Unterstützung in allen technischen Fragen dankbar. Ebenfalls dankbar bin ich Monika für ihre Hilfe bei jeglichen bürokratischen Angelegenheiten.

Abschließend danke ich noch meiner Familie und meinen Freunden, die mich während der Promotion begleitet haben. Ein besonderer Dank gilt meinen Eltern, die mich durch alle Höhen und Tiefen uneingeschränkt unterstützt haben.

Darmstadt, Juli 2018 Andreas Schmitt

Abstract

Simulations aid in many scientific and industrial applications. A general ambition for these simulations is to keep the time-to-solution as small as possible while maintaining a desired accuracy. Besides with high computational power, this can be achieved by employing multiple processing units with parallelization. Today's state of the art is the spatial parallelization which provides a very good parallel efficiency.

However, such a parallelization introduces communication and synchronization overheads leading to a maximal number of processing units which can be used efficiently. Applying a parallelization in time on top of the parallelization in space makes using more processing units possible. An issue of the parallel-in-time methods is their problem dependent efficiency. It tends to be generally bad for dominantly hyperbolic problems. The viscous Burgers equation, which for small viscosities falls into that category, is used to investigate two methods of parallelization in time.

First, a look is taken at the Adomian decomposition method (ADM) and possibilities of exploiting additional degrees of parallelism within this method. Its viability is questioned by comparing its discrete version (DADM) to the explicit Runge-Kutta method (ERK). The comparison shows similar restrictions regarding their maximal time step size for both methods. Furthermore, the DADM leads to larger errors with increasing order of accuracy compared to the ERK. However, discussing the parallelization within the DADM shows a reduction of the runtime complexity from quadratic to linear is possible. With this reduction in the runtime DADM seems to be a viable competitor to the ERK. This is especially true for high-order schemes, as fewer function evaluations have to be run serial. Increasing the order of accuracy is also embarrassingly easy with the DADM compared to the ERK.

The second method investigated in this thesis is the Parareal algorithm. Here, the focus lies on the potential of the implicit Runge-Kutta method with semi-Lagrangian advection (SLIRK) as the coarse solver for the Parareal algorithm. Its potential compared to using the explicit Runge-Kutta method (ERK) and the implicit-explicit Runge-Kutta method (IMEX) is tested with three different benchmarks. The comparison shows the ERK is in contrast to the other two methods not able to provide speedup potential with the chosen benchmarks. For advection dominated problems SLIRK performs better than IMEX due to its stability. The stability of SLIRK leads to speedup potential for a larger range of viscosities with the Parareal algorithm. Still, the instability of Parareal itself causes a decreasing potential with a decreasing viscosity. With an inviscid case the number of iterations to convergence for Parareal is too large to yield a reasonable speedup. An additional result worth mentioning is it was possible to show the importance of predicting the phase of the solution correctly for the convergence of Parareal.

Kurzfassung

Viele wissenschaftliche und industrielle Anwendungen werden von Simulationen unterstützt. Bei diesen Simulationen ist eine so gering wie mögliche Laufzeit bei einer vorgegebenen Genauigkeit gewünscht. Außer durch hohe Rechenleistung kann dies durch die Nutzung mehrerer Recheneinheiten mit Hilfe von Parallelisierung verringert werden. Derzeitiger Stand der Technik ist eine Parallelisierung im Raum, welche eine gute Effizienz aufweist.

Eine Parallelisierung erfordert jedoch unweigerlich Kommunikations- und Synchronisierungs-Overheads, welche zu einer maximalen Anzahl an Recheneinheiten führen, die effizient genutzt werden können. Ein Anwenden der Zeitparallelisierung zusätzlich zur räumlichen Parallelisierung kann die Effizienzgrenze allerdings weiter nach oben verschieben. Die Effizienz der Zeitparallelisierung ist jedoch abhängig von der Anwendung. Insbesondere Probleme, die vorwiegend hyperbolischer Natur sind, führen zu schlechter Effizienz. Anhand der viskosen Burgersgleichung, welche für kleine Viskositäten solch ein Problem ist, werden zwei Methoden der Zeitparallelisierung untersucht.

Zunächst wird die Möglichkeit einer Parallelisierung der Adomian decomposition method (ADM) untersucht. Da es sich bei der diskreten Version (DADM) der ADM um ein explizites Zeitschrittverfahren handelt, wird die Brauchbarkeit der DADM in einem Vergleich zur etablierten expliziten Runge-Kutta Methode (ERK) untersucht. Der Vergleich zeigt, dass beide Methoden einen sehr ähnlichen maximalen Zeitschritt zulassen. Bei ansteigender Verfahrensordnung führt die DADM zu einem größeren Fehler verglichen mit der ERK. Mit der gezeigten Parallelisierung der DADM ist es möglich die quadratische Laufzeitkomplexität auf eine lineare zu verringern. Dies führt dazu, dass die DADM mit der ERK konkurrieren kann. Insbesondere in Fällen mit hoher Verfahrensordnung ist die geringere Anzahl von Funktionsauswer-

tungen die seriell ablaufen ein Vorteil der DADM. Verglichen mit der ERK ist es sehr einfach die Verfahrensordnung der DADM zu erhöhen.

Die zweite untersuchte Methode ist der Parareal Algorithmus. Hierbei wird ein besonderes Augenmerk darauf gelegt, welches Potential die implizite Runge-Kutta Methode mit einer semi-Lagrangian Advektion (SLIRK) als Groblöser des Parareal Algorithmus besitzt. Hierfür wird ein Vergleich mit der expliziten Runge-Kutta Methode (ERK) und der impliziten-expliziten Runge-Kutta Methode (IMEX) auf Basis von drei Benchmarks durchgeführt. Dieser Vergleich zeigt, dass die ERK in den getesteten Fällen kein Speedup Potential besitzt. Bei kleinen Viskositäten, welche der Gleichung einen vorwiegend hyperbolischen Charakter geben, ist die SLIRK klar im Vorteil gegenüber der IMEX, da die SLIRK stabiler ist. Auf Grund dieser Stabilität führen mehr Viskositäten zu Speedup Potential mit dem Parareal Algorithmus. Ein Verringern der Viskosität resultiert jedoch in einem Anstieg der Zahl der Iterationen zur Konvergenz von Parareal durch die Instabilität von Parareal. Im Fall der Burgersgleichung ohne Viskosität kann kein Potential für Speedup gefunden werden. Zusätzlich ist zu Erwähnen, dass die korrekte Vorhersage der Phase der Lösung wichtig ist für eine schnelle Konvergenz von Parareal.

Contents

1 Introduction

1.1 Motivation

In today's research and industry numerical experiments with simulations gain increasing importance. They are especially interesting for, but not confined to, systems which can not be studied experimentally. Examples for such applications are planetary movements, weather prediction, or molecular interactions of pharmaceutics. Even in cases where experiments are possible simulations are a good alternative as they are often cheaper and faster than an experimental study. A general ambition for any kind of simulation is to have an as short as possible time-to-solution while maintaining a desired accuracy of the solution. This is especially challenging for problems where a solution with high accuracy is requested, as this usually tends to longer computational times. The desire for a short time-to-solution can be driven by different ambitions. In industry the goal is to reduce development costs and provide the possibility of computing and comparing multiple scenarios. In other cases a high accuracy is preferable, but a fixed maximal time-to-solution is the goal. Examples for such time critical problems are weather predictions [106] and early-warning systems for tsunamis to improve evacuation plans [17].

Over recent decades the requirements regarding accuracy were fulfilled by an increase of the spatial resolution [108]. The resulting rising requirement on computational power was supported and offset by the gain in clock-speed of the processing units for a long time. Since 2004 the gain in clock-speed has stagnated due to hitting physical limitations[1]. Today, further performance gains are no longer facilitated via the increasing clock-speed for free [100], but through utilizing multi-core systems with parallelism at instruction and

[1] See http://www.top500.org statistics

core levels. First steps in the direction of parallel computing on multi-core systems were already made in the 1960s with the first supercomputers, which were very expensive. In the 1990s grid computing became a cheaper alternative for some applications, where multiple desktop computers are combined to a multi-core virtual computer via the internet. After hitting the physical limitations regarding the clock-speed, commercial incentives drove the development of multi-core processors. This lead to affordable architectures for personal computers, also benefiting the performance and cost of supercomputers.

Parallelization of the spatial domain for simulations was the first self-evident approach. It is very well researched and de facto standard today. However, such a parallelization requires additional communication and synchronization leading to overheads. These overheads increase with the number of processing units up to a saturation limit, where adding more processing units is not beneficial anymore, as to much data hast to be sent. For some applications even the parallelization up to the saturation limit provides not enough speedup. Therefore, many approaches are currently under investigation to overcome such limitations. These approaches cover the range from new hardware (e.g. networking, new instruction sets, broader vector registers) to the software level (e.g. new algorithms for latency hiding, optimized network stacks, parallel-in-time methods).

In the present work, the focus lies on the software side with parallelization possibilities in the time domain to exploit resources beyond the spatial scalability. With the increasing amount of processing units in today's and tomorrow's high performance computing clusters, this is an important research field with many open questions, both practical and fundamental ones. Application-wise such a massive parallelization is in particular necessary for fields where short time scale phenomena have to be simulated for a comparably long time period, as it is the case e.g. in fluid dynamics. However, the current parallel-in-time methods show a very problem dependent efficiency. For problems classified as parabolic, like the heat transfer, very good efficiency can be achieved [95]. On the other hand, problems where the dominant part of the equation is classified as hyperbolic, as it is e.g. the case for fluid flows

with high Reynolds numbers, show small or even no speedups. These types of problems are investigated within this thesis and the effect of different time domain solvers on these is compared.

1.2 State of the Art

Fluid flow problems are tackled numerically in the field of computational fluid dynamics (CFD) by solving the Navier-Stokes equations (NSE), see e.g. [88, 107]. Acceleration in the time-to-solution is established by e.g. multigrid methods and parallelization in space with domain decomposition methods [20, 88]. A basic introduction in the domain decomposition topic is given e.g. by Dolean et al. [36].

As stated before, a parallelization in the time domain is desirable for the NSE, as long time periods have to be simulated with relatively small time steps. One possibility of parallelization is a parallelization within the time-stepping method. Examples for this are the Richardson extrapolation [81, 82] and the revisionist integral deferred correction (RIDC) method [28]. Both of these methods use multiple evaluations of cheap low-order time-stepping schemes to build a high-order time-stepping scheme. Since it is possible to carry out the low-order time-stepping scheme evaluations in parallel, overall a speedup is gained compared to running an expensive high-order method. Within this thesis the possibilities of a parallelization of the discrete Adomian decomposition method, based on its continuous version proposed by Adomian [4–8], is investigated. However, a parallelization with these methods, where parallelization is only gained by increasing the order of accuracy, is only possible on a small scale, if the desired accuracy order is not very large.

Large scale parallelization in time is possible with parallel-in-time (PinT) methods[2]. An extensive review over the first 50 years of parallel-in-time methods is provided by Gander in [47]. According to this review, the first method introduced to parallelize the temporal domain was published by Nievergelt in [75]. He foresaw the trend towards multi-core architectures and the need

[2] On http://www.parallel-in-time.org resources and information about this topic are gathered

for parallelization already in 1964. The idea of the algorithm is to split the time domain into multiple parts. Then a rough prediction is done to get an estimation for the solutions of the parts. These estimated solutions and some points in their vicinity are used as initial conditions for parallel computations with an accurate method. Finally, interpolation is used to best match the correct solution with the array of curves computed.

The PinT method, which increased the attention to parallel-in-time methods in general, is the Parareal algorithm by Lions et al. [71]. For this algorithm the time domain is split in multiple time-slices as well. A cheap coarse solver is used to estimate the solutions at all time-slices, which then are used as the initial condition for an expensive fine solver in parallel. A continuous solution is then found by iteration. It is applied to a broad range of applications, e.g. financial mathematics [13], quantum chemistry [73], biology [18], and engineering [44, 51, 72, 91, 102, 103]. Its popularity is partially based on the fact that it is a non-intrusive algorithm, which makes it easily applicable to existing solvers. Parareal is the algorithm of choice for this thesis because of it being non-intrusive and because it is the basis of most of the newer, more sophisticated PinT methods.

Parareal inspired the parallel implicit time-integration algorithm (PITA) by Farhat & Chandesris in [42], which is shown to be equivalent to Parareal for linear cases [52]. PITA was applied to and shows some potential for fluid, structure, and fluid-structure problems [42]. The results of the structure problems did show a lack in efficiency for second-order hyperbolic equations. Gander & Vandewalle [53] have shown that Parareal can be written as a multiple-shooting and a multigrid in time method. In its multigrid in time form, it is the special case of the multigrid reduction in time (MGRIT) algorithm by Friedhoff et al. [45] with two grids. The MGRIT algorithm is another non-intrusive PinT approach, which transfers the idea of the spatial multigrid method to the time domain. Two other methods combining the multigrid method with time parallelization are the space-time multigrid (STMG) [59] and the space-time concurrent multigrid waveform relaxation (WRMG) [60, 105], which are both limited to parabolic partial differential equations.

Another PinT method derived from the Parareal algorithm employing multigrid techniques is the parallel full approximation scheme in space and time (PFASST) by Emmett & Minion in [41]. In this algorithm the direct solvers are replaced by spectral deferred corrections (SDC) [38]. With this algorithm it is possible to reach higher efficiency than with Parareal. However, applying this algorithm to already existing code is cumbersome, as it is very intrusive. This was the reason for not considering it for the present work, because the long term goal of this thesis is to apply a parallelization in time to the in-house flow solver FASTEST.

None of these PinT methods has proven to be very efficient for high Reynolds number flows, which are problems with dominantly hyperbolic terms. Multiple authors have shown for the Parareal algorithm that the problems occurring with (dominantly) hyperbolic equations are mainly related to stability problems of the Parareal algorithm [34, 46, 53, 96]. These findings are supported by results of numerical experiments e.g. in [44, 98]. Modifications to the Parareal algorithm are able to reach stabilization for some hyperbolic problems. A stabilization for first and second-order hyperbolic systems is proposed in [34]. Within this modification a projection method is used to ensure conservation of a system specific Hamiltonian (energy) for all the intermediate solutions. A different approach for stabilization is taken in [25]. Here, a coarse solver is employed which is based on reduced basis methods leading besides the stabilization to faster convergence of the Parareal algorithm. A modification was also proposed for the PITA algorithm to increase its efficiency for second-order linear hyperbolic systems [43]. An extension to non-linear problems is introduced in [30] and tested with non-linear structural dynamics problems. The Krylov subspace based projection used for PITA was also applied to the Parareal algorithm to stabilize it for linear hyperbolic problems [52, 86].

Another method using Krylov subspace methods is the ParaExp method which was proposed in [48]. It is able to efficiently parallelize linear initial value problems including those of hyperbolic type making use of exponential solvers. Recently, a non-linear variant of the method was proposed and analyzed in [49]. Kooij et al. [65] show realistic speedups for the advection-diffusion and the viscous Burgers equation with this method. In their results

in [66] they were able to extend their exponential time integration method to the incompressible Navier-Stokes equations hinting that with ParaExp parallel efficiency in time is possible.

The referenced research has shown the relation between the efficiency problems of Parareal and its stability, which is very dependent on the chosen coarse solver. Because of this, the benefits of using a semi-Lagrangian implicit Runge-Kutta method as a coarse solver is investigated in this thesis. Semi-Lagrangian methods are widely used in the geophysical fluid dynamics community [97]. Their popularity is due to their large stability region making large time steps possible. Possible benefits of using this method were already suggested by [31, 80], however, no investigation was carried out up to now.

Instead of running the investigations throughout this thesis with the complex NSE, a simpler equation showing comparable mathematical properties is used for practical reasons. This equation is the viscous Burgers equation, first introduced by Bateman in [16]. It is a good starting point for testing numerical schemes applied to the NSE. Burgers studied the equations extensively with the purpose of deriving a statistical theory of turbulent fluid motion, modeled after the classical statistical mechanics as applied to the kinetic theory of gases [21, 22]. The works of Cole [29] and Hopf [57] have shown the lack of a mechanism to mix features of the initial data to generate new randomness in the Burgers equation, which would be necessary to yield a theory of turbulent fluid motion. Nevertheless, important parts of the NSE are present in the Burgers equation making it a viable choice for the presented research.

1.3 Research Question

The governing question of this work is whether it is possible to accelerate fluid flow computations with parallelization in time. This question can be split in the following parts.

- Is a parallelization of the discrete Adomian decomposition method competitive to established serial time-stepping schemes?

- Is it possible to stabilize the Parareal algorithm with a very stable coarse solver?
- Is the instability of Parareal with non-linear problems dependent on phase error of the wave propagation, as it is the case with linear problems [85]?

1.4 Structure of the Thesis

The further parts of the thesis are structured as follows. In Chap. 2 the mathematical and physical basics are provided. These basics contain a short review of the classification of partial differential equations. In addition, the relation between the Navier-Stokes equations and the viscous Burgers equation is shown. The discretization and parallelization methods, which are applied to the Burgers equation in this thesis, are given in Chap. 3. In this chapter the SLIRK-Parareal algorithm is introduced, which is an adaption of the standard Parareal algorithm, necessary to apply the second-order semi-Lagrangian implicit Runge-Kutta (SLIRK) method as the coarse solver. The implementation of the numerical methods is verified in Chap. 4. In Chap. 5 a parallelization possibility within the discrete Adomian decomposition method is introduced. In addition, its viability is investigated by comparing it to the Runge-Kutta method. The investigation of the Parareal algorithm considering different coarse solvers is carried out in Chap. 6. A special focus is laid on the effects of the semi-Lagrangian implicit Runge-Kutta method as a coarse solver. Finally, a conclusion and an outlook are provided in Chap. 7.

2 Mathematical and Physical Basics

In the introduction it was already stated that within this thesis the viscous Burgers equation will be used to investigate different numerical schemes. Furthermore, it was stated that hyperbolic equations pose a big challenge for parallel-in-time methods with respect to gaining an efficient parallelization. Both, the classification of partial differential equations (PDEs) into elliptic, parabolic, and hyperbolic as well as the reasoning behind the choice of the viscous Burgers equation are explained in this chapter. Due to using only academic examples within this thesis, no units are used with the physical quantities. Any consistently scaled unit system could be used here.

2.1 Classification of Partial Differential Equations

Partial differential equations can be classified by multiple criteria, as discussed in detail e.g. by Durran [37]. The most simple distinction is the order of the PDE. It is equal to the order of the highest-order partial derivative. PDEs of same order can be further distinguished as linear, or non-linear considering its unknown. A subgroup of the non-linear PDEs is build by the quasi-linear PDEs. This subgroup consists of all non-linear PDEs of order p which are linear in the derivatives of order p. This class of PDEs has the advantage of solving them by easily generalizing solution techniques for linear PDEs.

Additionally, second-order PDEs can be classified as hyperbolic, parabolic, and elliptic. For this classification the notation and definition of Dahmen & Reusken in [33] is followed here. Considering the general second-order linear differential operator

$$\mathcal{L}u = a_{11}\frac{\partial^2 u}{\partial t^2} + 2a_{12}\frac{\partial^2 u}{\partial t \partial x} + a_{22}\frac{\partial^2 u}{\partial x^2} + b_1\frac{\partial u}{\partial t} + b_2\frac{\partial u}{\partial x} + cu \tag{2.1}$$

in two variables $(t, x) \in \Omega$, where Ω is the domain of interest, the coefficients can be written in the symmetric matrix

$$\mathbf{A}(t,x) = \begin{pmatrix} a_{11}(t,x) & a_{12}(t,x) \\ a_{12}(t,x) & a_{22}(t,x) \end{pmatrix} . \tag{2.2}$$

The differential operator $\mathcal{L}$ is called elliptic, if the eigenvalues of $\mathbf{A}$ are non-zero and have the same sign for all $(t, x) \in \Omega$. If the eigenvalues of $\mathbf{A}$ have different signs and are non-zero for all $(t, x) \in \Omega$, then $\mathcal{L}$ is called hyperbolic. In case of exactly one eigenvalue of $\mathbf{A}$ being zero and the extended matrix

$$\begin{pmatrix} a_{11}(t,x) & a_{12}(t,x) & b_1(t,x) \\ a_{12}(t,x) & a_{22}(t,x) & b_2(t,x) \end{pmatrix} \tag{2.3}$$

having rank two for all $(t, x) \in \Omega$ the differential operator $\mathcal{L}$ is called parabolic. This definition can also be used to classify differential operators in higher dimensions.

Because of $\det(\mathbf{A}) = \lambda_1 \lambda_2$, with λ_1 and λ_2 being the eigenvalues of $\mathbf{A}$, often the classification

- elliptic if $a_{12}^2 - a_{11}a_{22} < 0$
- parabolic if $a_{12}^2 - a_{11}a_{22} = 0$
- hyperbolic if $a_{12}^2 - a_{11}a_{22} > 0$

can be found in literature. The origin of the names lies in the similarity to the equation

$$a_{11}x^2 + 2a_{12}xy + a_{22}y^2 = c \, , \tag{2.4}$$

which describes the cut between a cone and a plane. The cuts possible are either an ellipse, a parabola, or a hyperbola, where the type of cut is also determined by the given criteria.

The classification as hyperbolic is also possible for PDEs of arbitrary order [84]. Based on the corresponding definition any first-order PDE is hyperbolic.

2.2 Navier-Stokes and Burgers' Equations

Fluid dynamics are described mathematically by balance laws for mass, momentum, and energy. These balance laws can be mathematically represented by PDEs equating the change over time of the conserved quantity with its change due to advection, diffusion, sources, and sinks. Consequently, computational fluid dynamics models are composed of respective systems of partial differential equations [88, 107].

The basic equations describing the conservation of momentum in fluid dynamics are the Navier-Stokes equations. A related equation, not fully suited for describing general fluid dynamics is the viscous Burgers equation. The Burgers equation is used in this thesis to examine the behavior of the different numerical methods. Because of the relation between the two equations results gathered for the Burgers equation provide hints on the behavior of the numerical schemes applied to the Navier-Stokes equations.

2.2.1 Compressible Navier-Stokes Equations

In the literature the term Navier-Stokes equations is often used to refer to both the conservation of mass and momentum. This is adopted for this thesis. In their compressible form the system of equations is closed by the equation for the conservation of energy and equations of state. These additional equations are not discussed here, because they are not of interest for the connection to the Burgers equation.

For the conservation of mass it is assumed that mass is neither created, nor destroyed within the observed system. This yields the equation

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0 \,, \tag{2.5}$$

which is the conservation in mass in its differential form, where t denotes time, ρ density, and $\mathbf{u}$ the velocity vector consisting of the velocities in each spatial direction. The equation is also referred to as the continuity equation.

In case of the conservation of momentum external forces can act as sources and sinks. Additionally, the flux of momentum to and from the system has to be accounted for. Therefore, the differential form reads

$$\frac{\partial (\rho \mathbf{u})}{\partial t} + \nabla \cdot (\rho \mathbf{u} \otimes \mathbf{u}) = \nabla \cdot \boldsymbol{\tau} + \rho \mathbf{F} \,, \tag{2.6}$$

where $\mathbf{F}$ is the collection of external accelerations induced by e.g. gravity or magnetic forces and $\boldsymbol{\tau}$ is the Cauchy stress tensor. Here, Newtonian fluids, i.e. fluids where the viscous stresses are linearly proportional to the local strain rate, are considered. For these fluids the entries of the Cauchy stress tensor are

$$\tau_{i,j} = \mu \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) - \left(\frac{2}{3} \mu \nabla \cdot \mathbf{u} + p \right) \delta_{i,j} \,, \tag{2.7}$$

with the dynamic viscosity μ, the pressure p and the Kronecker delta $\delta_{i,j}$.

2.2.2 Incompressible Navier-Stokes equations

For specific cases of fluid flow the assumption of incompressibility ($\mathrm{D}\rho/\mathrm{D}t = 0$) is valid. In practice a general rule of thumb for incompressibility is $Ma < 0.3$, with the Mach number

$$Ma = \frac{U}{c} \,, \tag{2.8}$$

where U is the characteristic flow velocity and c the speed of sound [107]. The compressible Navier-Stokes equations (2.5) and (2.6) then simplify to the incompressible Navier-Stokes equations

$$\nabla \cdot \mathbf{u} = 0 \tag{2.9}$$

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\, \mathbf{u} = -\frac{1}{\rho} \nabla p + \nu \nabla^2 \mathbf{u} + \mathbf{F} \,, \tag{2.10}$$

where ν denotes the kinematic viscosity. Equation (2.9) is a hyperbolic linear PDE of first-order and (2.10) is a parabolic quasi-linear PDE of second-order. In case of very large Reynolds numbers

$$Re = \frac{UL}{\nu} \gg 1 \,, \tag{2.11}$$

where L denotes a characteristic length and U a characteristic velocity, the influence of the diffusive term diminishes. Therefore, the PDE (2.10) behaves like a hyperbolic PDE [37]. For realistic industrial and environmental flows this case is quite common [107].

2.2.3 Burgers' Equation

Applying the additional assumption of a pressure free fluid ($p = 0$) and no external accelerations ($\mathbf{F} = 0$) to the incompressible Navier-Stokes equations leads to the viscous Burgers equation

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu\frac{\partial^2 u}{\partial x^2} \tag{2.12}$$

which in contrast to (2.9) and (2.10) can be investigated in one spatial dimension. Since the pressure as an unknown variable is eliminated, the equation originating from the conservation of momentum suffices to determine a solution.

This equation was introduced by Bateman in [16] to demonstrate the possible discontinuity in the equations of motion of a viscous fluid when the viscosity approaches zero. Burgers then studied the equation extensively, see [22]. He originally intended to find a statistical theory of turbulent flow in which he was unsuccessful. Nevertheless, his attempts showed the combination of dissipative and non-linear inertia terms in the equations of motion being the reason for essential features of the spectrum of turbulence. The Burgers equation yields a simple model to investigate such an interaction.

The similarity between the Burgers and the incompressible Navier-Stokes equation can be seen as well in their properties regarding their classification. The Burgers equation is also a parabolic quasi-linear PDE of second-order, which behaves like a hyperbolic PDE for $\nu \to 0$. In the context of parallel-in-time methods the transition to a hyperbolic PDE is worth studying, as parallel-in-time methods are not able to provide significant speedups for hyperbolic equations, see e.g. Staff & Rønquist [96]. The advantage of the Burgers equation is the reduction of the complexity of the Navier-Stokes

equations by focusing on the part, which is interesting for parallel-in-time methods.

The Burgers equation with $\nu = 0$

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = 0 \tag{2.13}$$

is referred to as the inviscid Burgers equation. It is able to describe the development of discontinuities, like shock waves, which makes it a prototype for conservation equations with this feature.

2.3 Cole-Hopf Transformation

A solution to the Burgers equation can be found not only within the family of solutions proposed by Bateman in [16], but also in the general case by employing the Cole-Hopf transformation. This transformation was published independently by Cole [29] and Hopf [57]. Using the transformation

$$u(t,x) = -2\nu\frac{\partial \log(\Phi(t,x))}{\partial x} \tag{2.14}$$

the viscous Burgers equation (2.12) can be rewritten as the linear heat equation

$$\frac{\partial \Phi}{\partial t} = \nu\frac{\partial^2 \Phi}{\partial x^2}\ , \tag{2.15}$$

ignoring a necessary time-dependent factor, which is irrelevant for the solution u. Assuming the initial condition $u(0,x) = u_0(x)$ for the Burgers equation is given, the initial condition for (2.15) is

$$\Phi(0,x) = \exp\left(-\frac{1}{2\nu}\int_0^x u_0(\xi)\ \mathrm{d}\xi\right)\ . \tag{2.16}$$

The lower bound of the integration is arbitrary, as the integration constant has no influence on the final solution. Solving (2.15) yields $\Phi(t,x)$, which can be transformed back to $u(t,x)$ to get the solution to the viscous Burgers equation.

The solution to (2.15) is given by the expression

$$\Phi(t,x) = \frac{1}{\sqrt{4\pi\nu t}} \int_{-\infty}^{\infty} \exp\left(-\frac{(x-\xi)^2}{4\nu t}\right) \Phi(\xi,0)\, \mathrm{d}\xi \,.$$

Using the Cole-Hopf transformation on the result leads to

$$\begin{aligned} u(t,x) &= -2\nu \frac{\partial \log(\Phi)}{\partial x} \\ &= \frac{\displaystyle\int_{-\infty}^{\infty} \frac{(x-\xi)}{t} \exp\left(-\frac{(x-\xi)^2}{4\nu t}\right) \Phi(\xi,0)\, \mathrm{d}\xi}{\displaystyle\int_{-\infty}^{\infty} \exp\left(-\frac{(x-\xi)^2}{4\nu t}\right) \Phi(\xi,0)\, \mathrm{d}\xi} \,, \end{aligned} \tag{2.17}$$

which is the analytical solution to the viscous Burgers equation.

Under the assumption that the initial condition $u_0(x)$ is integrable, it is possible to compute the solution at any time t with (2.17). The downside is the effortful computation. This is especially true for cases, where the initial condition (2.16) evaluates to huge or vanishingly small values. For $\nu \to 0$ the convergence to the solution of the inviscid Burgers equation given by the method of characteristics can be shown, as it is presented e.g. by Ablowitz [3].

3 Numerical Basics

Solving the Burgers equation numerically is not necessary in most cases, as there exists the analytical solution revised in Sec. 2.3. However, since numerical schemes are required to solve the Navier-Stokes equations, the connection between the Burgers equation and the Navier-Stokes equations makes testing the schemes on the Burgers equation worthwhile. This chapter contains the numerical basics for the work presented in this thesis. Already well-known concepts are reviewed in short and less known and important ones, like the semi-Lagrangian method, the discrete Adomian decomposition, and the Parareal algorithm, are explained in more detail.

3.1 Eulerian and Lagrangian Framework

To understand the semi-Lagrangian method, it is necessary to know the difference between the Eulerian and Lagrangian framework, as described e.g. in [15, 67]. The viscous Burgers equation as given in Sec. 2.2.3 is written in its Eulerian formulation. With the definition of the velocity

$$\frac{\mathrm{d}x(t)}{\mathrm{d}t} = u(t, x(t))$$

and the total derivative

$$\frac{\mathrm{d}}{\mathrm{d}t} = \frac{\partial}{\partial t} + \frac{\mathrm{d}x}{\mathrm{d}t}\frac{\partial}{\partial x} ,$$

equation (2.12) can be rewritten in its Lagrangian form

$$\frac{\mathrm{d}u}{\mathrm{d}t} = \nu \frac{\partial^2 u}{\partial x^2} . \tag{3.1}$$

From an analytical standpoint the difference between the two is the notation of the time derivative, which is on the one hand the total derivative and on

the other hand expanded to its partial parts, the partial time derivative and the advection term.

The distinction from a numerical standpoint is bigger. The Eulerian and the Lagrangian formulation implicate different numerical grid behavior. Within an Eulerian framework the Eulerian formulation is usually used with an Eulerian grid, i.e. the grid is fixed over time in the whole observed domain and the observed entity moves through the grid. This is e.g. typically used for fluid flow problems. On the contrary, the Lagrangian formulation is mostly paired with a Lagrangian grid to build a Lagrangian framework, i.e. the grid moves over time as it is fixed to and follows the observed entity. An application for this scenario would naturally be the simulation of structures.

3.2 Semi-Lagrangian Method

Besides the Eulerian and Lagrangian framework (see Sec. 3.1) there is also the semi-Lagrangian method. It is frequently and successfully used in geophysical fluid dynamics in order to compute advection dominated problems with larger time step sizes [97]. This property of the semi-Lagrangian method leads to investigating its potential with parallel-in-time methods. The semi-Lagrangian method used in this work was already described in [89] and is reviewed in this section. For further information on this topic, see e.g. [19, 37].

The basic idea of the semi-Lagrangian method is to mix parts of the Eulerian and Lagrangian framework. The equation of the simulation is used in its Lagrangian formulation, but instead of pairing it with a Lagrangian grid, an Eulerian grid is adopted. Therefore, the particle movement is calculated in the Lagrangian framework and the simulation data is stored on an Eulerian grid. The values of the particle grid are interpolated onto the Eulerian grid within the calculation.

With this method the Lagrangian trajectory is resolved, which ensures the complete coverage of the numerical domain of dependence by the physical domain of dependence. This is a necessary condition for unconditional stability (independent of time step size) with respect to the advective term. If, in addition, a stable trajectory calculation is used, then the semi-Lagrangian

method is not restricted by the Courant Friedrichs Lewy (CFL) condition [32] for advection dominated problems. It is only restricted to accuracy conditions.

Using a simple time discretization for the total derivative yields

$$\frac{u(t_{n+1}, x(t_{n+1})) - u(t_n, x(t_n))}{\Delta t} \,, \tag{3.2}$$

where a time step width Δt is chosen to discretize the time domain, such that $t_{n+1} = t_n + \Delta t$. The velocity at the new time point t_{n+1} is stored at the grid points defined as $x(t_{n+1})$. These grid points are commonly denoted as the arrival points x_a. The points defined as $x(t_n)$ at the current time point t_n now have to be estimated. These are called departure points x_d. The Lagrangian trajectory ordinary differential equation (ODE)

$$\frac{\mathrm{d}x(t)}{\mathrm{d}t} = u(t, x(t)) \tag{3.3}$$

is used to calculate the departure points. Integrating (3.3) from t_n to t_{n+1} yields

$$x_a - x_d = \int_{t_n}^{t_{n+1}} u(t, x(t)) \, \mathrm{d}t \,. \tag{3.4}$$

By approximating the integral with the midpoint rule

$$x_a - x_d = u(t_{n+1/2}, x_m)\Delta t \tag{3.5}$$

the midpoints x_m at $x(t_{n+1/2})$ are introduced. The intermediate velocity $u(t_{n+1/2}, x_m)$ is then computed by extrapolation. It is important to chose an extrapolation which ensures stability [37]. Within this thesis the stable extrapolation two-time level scheme (SETTLS) from Hortal [58] is chosen. With this scheme the intermediate velocity reads

$$u(t_{n+1/2}, x_m) \approx \frac{1}{2}(2u(t_n, x_d) - u(t_{n-1}, x_d) + u(t_n, x_a)) \,, \tag{3.6}$$

where also information from the previous time step t_{n-1} is used. Combining (3.5) and (3.6) provides a non-linear implicit equation for x_d. This equation can be solved using an iterative scheme

$$x_d^{k+1} = x_a - \frac{\Delta t}{2}(2u(t_n, x_d^k) - u(t_{n-1}, x_d^k) + u(t_n, x_a)) \,, \tag{3.7}$$

where k denotes the iteration index and for the initial guess $x_d^0 = x_a$ is used.

Since the x_d^k are typically not coinciding with the grid points, an interpolation is applied. In this thesis a second-order bi-linear interpolation with respect to the nearest grid points [97] is employed.

The iteration can be rewritten as

$$x_d^{k+1} = x_a - \frac{\Delta t}{2} u(t_n) - \frac{\Delta t}{2} (2u(t_n) - u(t_{n-1}))_* \ ,$$

where $(\cdot)_*$ denotes the values of a field which are interpolated to the departure points. With this notation and $u^n = u(t_n)$ it is possible to write (3.2) as

$$\frac{u^{n+1} - u_*^n}{\Delta t} \ ,$$

where all unknowns are eliminated. The interpolation to the departure points is done with fourth-order accuracy (bi-cubic interpolation). This ensures together with the bi-linear interpolation of the velocities an overall second-order scheme with respect to the advection [77].

An overall first-order scheme can be achieved by approximating the integral in (3.4) with

$$x_a - x_d = u(t_n, x_a) \Delta t \ . \tag{3.8}$$

The interpolation to the departure points is still done with the bi-cubic interpolation within this thesis.

It has to be remarked, that numerical solutions to (3.2) do not necessarily conserve the domain integral of a passive tracer [37]. This means there typically is no mass conservation with this approach. In the literature there are multiple suggestions on how to ensure mass conservation, e.g. [68, 70, 74, 76, 79, 110]. For the purpose of this thesis, such alterations are not necessary, since this formulation is used in combination with the Parareal algorithm and with this combination the corresponding errors do not matter, as shown in Section 6.6.3.

3.3 Spatial Discretization

Independent of the formulation used for the Burgers equation, it is necessary to discretize space and time to calculate a solution numerically. The Burgers equation is discretized with the method of lines, which means the PDE is discretized in space to gain a system of ordinary differential equations in time. Since the focus of the thesis lies on the time integration methods applied on the ODEs, an as accurate as possible spatial discretization is desired.

A highly accurate discretization is provided by the spectral method. It is a method from the family of series-expansion methods. Other well-known methods of this family are the finite-element and discontinuous Galerkin methods. An in-depth explanation of the spectral method is given e.g. by Durran in [37]. The main points are sketched here.

Following the notation of [37], the PDE to be solved on the spatial domain Ω is given by

$$\frac{\partial u}{\partial t} + f(u) = 0 \,, \tag{3.9}$$

where f contains spatial derivatives of u, with the necessary initial and boundary conditions present. The spatial dependence of u is then approximated by a linear combination of a finite number M of known expansion functions. The approximation $\bar{u}$ then reads

$$\bar{u}(x,t) = \sum_{i=1}^{M} u_i(t)\varphi_i(x) \,. \tag{3.10}$$

The φ_i denote the known expansion functions, which satisfy the boundary conditions of the given problem. Since the expansion functions are known, the new unknowns are the $u_i(t)$. In general it is not possible to compute the unknowns such that (3.10) is an exact solution to (3.9). Therefore, the aim is to minimize the residual

$$R(\bar{u}) = \frac{\partial \bar{u}}{\partial t} + f(\bar{u}) \,. \tag{3.11}$$

This minimization can be carried out with three different strategies, each leading to a coupled system of ODEs for the time-dependent unknowns $u_i(t)$. The three strategies are

1. to minimize the square of the L_2-norm of the residual,
2. the collocation, where the residual is required to be zero at a discrete set of points, and
3. the Galerkin approximation, where the residual has to be orthogonal to each of the expansion functions.

If a problem like (3.9) is considered, then both the L_2-norm and Galerkin strategies are equivalent. These are also the two strategies which can be used for the spectral method. Here, the Galerkin approximation is used for the further explanation. The Galerkin requirement

$$\int_\Omega R(\bar{u})\varphi_i \,\mathrm{d}x = 0 \qquad \text{for all } i = 1, \ldots, M \tag{3.12}$$

leads to the system of ODEs

$$\sum_{j=1}^{M} m_{i,j} \frac{\mathrm{d}u_j}{\mathrm{d}t} = -\int_\Omega \left[f\left(\sum_{j=1}^{M} u_j \varphi_j \right) \varphi_i \right] \mathrm{d}x \qquad \text{for all } i = 1, \ldots, M \ , \tag{3.13}$$

where $m_{i,j} = \int_\Omega \varphi_j \varphi_i \,\mathrm{d}x$ are the components of the mass matrix, following the terminology from continuum mechanics. Employing the same approximation procedure to the given initial conditions provides the initial conditions for the system of ODEs.

In contrast to the finite-element method with its piecewise polynomial ansatz functions, the spectral method uses expansion functions which form an orthogonal set. Due to this, it is possible to rewrite (3.13) as

$$\frac{\mathrm{d}u_i}{\mathrm{d}t} = -\frac{1}{m_{i,i}} \int_\Omega \left[f\left(\sum_{j=1}^{M} u_j \varphi_j \right) \varphi_i \right] \mathrm{d}x \qquad \text{for all } i = 1, \ldots, M \ , \tag{3.14}$$

as all combinations of $\varphi_i \varphi_j$ from (3.13) are equal to zero for $i \neq j$. The orthogonal expansion functions chosen for this thesis are the terms of the Fourier series. Therefore, all boundary conditions have to be periodic.

With this method it is possible to apply linear operators directly to the Fourier modes leading to errors which are in the range of machine precision.

In theory it would also be possible to compute the non-linear terms with high precision, however, this has a quadratic complexity. This complexity is circumvented by a pseudo-spectral approach [14, 54]. The non-linearities are calculated by transforming the multiplicand and multiplier to the physical space before carrying out the multiplication node-wise with this approach.

The node-wise calculation may lead to spurious modes in the spectral representation. An anti-aliasing technique is employed to counteract these modes. For this technique a higher resolution is used in the physical space than in the spectral space, such that during the conversion to spectral space high modes are truncated, see e.g. Press et al. [78]. The employed anti-aliasing technique of this thesis is a $2/3$ anti-aliasing rule, where the spectral resolution is two-thirds of the physical resolution.

Definite Helmholtz equations, having a positive definite spectrum, as in Eqs. (3.16) and (3.18) can be solved accurately and efficiently with element-wise vector-vector multiplication in spectral space [93, 101], which is another advantage of the chosen spatial discretization for this work.

3.4 Time-Stepping Schemes

In addition to the discretization in space, a discretization in time has to be carried out. Here, an equidistant discretization of the domain is used. Multiple different time-stepping schemes are used to numerically integrate in the time domain. A short overview on the time-stepping schemes used in this thesis is provided in the following.

3.4.1 Explicit Runge-Kutta

The explicit Runge-Kutta method (ERK) is well-known [69]. Although the term often refers just to the fourth-order version, it is possible to build schemes

for an arbitrary order p. The method is used to solve $\mathrm{d}u/\mathrm{d}t = f(t,u)$ and reads in short

$$
\begin{aligned}
u^{n+1} &= u^n + \Delta t \sum_{i=1}^{s} b_i k_i \\
k_1 &= f(t_n, u^n) \\
k_i &= f\left(t_n + c_i \Delta t, u^n + \Delta t \sum_{j=1}^{i-1} a_{i,j} k_j \right) , \qquad \text{for } i = 2, \ldots, s \, .
\end{aligned}
$$

The coefficients $a_{i,j}, b_i, c_i$ appearing in the method are typically arranged in a Butcher tableau [23]

$$
\begin{array}{c|cccc}
0 & & & & \\
c_2 & a_{2,1} & & & \\
\vdots & \vdots & \ddots & & \\
c_s & a_{s,1} & \cdots & a_{s,s-1} & \\
\hline
 & b_1 & \cdots & b_{s-1} & b_s
\end{array} \, .
$$

The conditions necessary to determine the coefficients for an order p scheme can be found in the literature [e.g. 69]. Because of these conditions the order p of an s-stage ERK method is bound by $s \geq p$. In case of $p \geq 5$ the inequality $s > p$ holds [24]. These constraints cause the number of function evaluations to be equal to, or higher than the order of the scheme p, as each stage contains one function evaluation.

The ERK method has a limited stability region, as it is an explicit scheme. The time step is bound by the CFL condition, i.e. the time step has to be chosen in a way to ensure that the numerical domain of dependence covers the whole physical domain of dependence. In the case of Burgers' equation the velocities defining the physical domain of dependence can be driven by advection, or by diffusion. This leads to the maximal time step $\Delta t_{\max} \propto \Delta x$ and $\Delta t_{\max} \propto \Delta x^2$, respectively.

3.4.2 Implicit Runge-Kutta

The Runge-Kutta method can also be formulated in an implicit way [69]. The implicit Runge-Kutta method (IRK) to solve $\mathrm{d}u/\mathrm{d}t = f(t,u)$ is

$$u^{n+1} = u^n + \Delta t \sum_{i=1}^{s} b_i k_i$$

$$k_i = f\left(t_n + c_i \Delta t, u^n + \Delta t \sum_{j=1}^{s} a_{i,j} k_j\right), \qquad \text{for } i = 1, \ldots, s$$

and is only slightly different than the explicit method. In the Butcher tableau

$$\begin{array}{c|ccc} c_1 & a_{1,1} & \cdots & a_{1,s} \\ \vdots & \vdots & \ddots & \vdots \\ c_s & a_{s,1} & \cdots & a_{s,s} \\ \hline & b_1 & \cdots & b_s \end{array}$$

the difference becomes obvious. For the implicit schemes there are entries for $a_{i,j}$ on the diagonal and, entries above the diagonal are possible. Methods, where the entries above the diagonal are zero, are also called diagonally implicit Runge-Kutta method (DIRK) [56].

With this method being implicit it reduces the stability restrictions on the time step size. The choice of larger time steps, however, introduces larger phase and amplitude errors [37].

3.4.3 Implicit-Explicit Runge-Kutta

Besides the ERK and IRK there is also the implicit-explicit Runge-Kutta method (IMEX). For more detailed information on this topic than the ones given below, the reader is referred to e.g. [10, 64].

Generally different stability restrictions can be found within one equation. The idea of IMEX is to discretize each term of the equation with a scheme addressing its specific restrictions as good as possible. With this method it is then possible to discretize stiff, linear terms $f(t,u)$ with an implicit s-stage DIRK scheme and not so stiff, non-linear terms $g(t,u)$ with an explicit $(s+1)$-stage scheme, where $\sigma = s+1$. Ascher et al. [10] introduced an algorithm

to compute a time step from t_n to $t_{n+1} = t_n + \Delta t$ with IMEX following the algorithm:

Set

$$\hat{k}_1 = g(t_n, u^n) \, .$$

For $i = 1, \ldots, s$ do:

$$k_i = f\left(t_n + c_i \Delta t, u^n + \Delta t \sum_{j=1}^{i} a_{i,j} k_j + \Delta t \sum_{j=1}^{i} \hat{a}_{i+1,j} \hat{k}_j \right) \, ,$$

$$\hat{k}_{i+1} = g\left(t_n + \hat{c}_i \Delta t, u^n + \Delta t \sum_{j=1}^{i} a_{i,j} k_j + \Delta t \sum_{j=1}^{i} \hat{a}_{i+1,j} \hat{k}_j \right) \, .$$

Evaluate

$$u^{n+1} = u^n + \Delta t \sum_{j=1}^{s} b_j k_j + \Delta t \sum_{j=1}^{\sigma} \hat{b}_j \hat{k}_j \, .$$

The constants are listed in the Butcher tableaux

$$\begin{array}{c|ccc} 0 & & & \\ \hat{c}_2 & \hat{a}_{2,1} & & \\ \vdots & \vdots & \ddots & \\ \hat{c}_\sigma & \hat{a}_{\sigma,1} & \cdots & \hat{a}_{\sigma,\sigma-1} \\ \hline & \hat{b}_1 & \cdots & \hat{b}_{\sigma-1} \quad \hat{b}_\sigma \end{array} \quad \text{and} \quad \begin{array}{c|ccc} c_1 & a_{1,1} & & \\ \vdots & \vdots & \ddots & \\ c_s & a_{s,1} & \cdots & a_{s,s} \\ \hline & b_1 & \cdots & b_s \end{array}$$

for the explicit and implicit scheme, respectively. If both schemes are of same order p, IMEX is also of order p.

In case of the viscous Burgers equation used in this thesis, the implicit scheme is used for the diffusive term and the explicit scheme for the advective term. This leads to definite Helmholtz problems for the implicit equations. As mentioned in Sec. 3.3 such a problem can be solved efficiently with the chosen spatial discretization. Using the first-order IMEX

$$u^{n+1} = u^n + \Delta t \left(u^n \frac{\partial u^n}{\partial x} + \nu \frac{\partial^2 u^{n+1}}{\partial x^2} \right) \tag{3.15}$$

as an example, the definite Helmholtz problem

$$\left(1 - \nu\Delta t \frac{\partial^2}{\partial x^2}\right) u^{n+1} = u^n + \Delta t u^n \frac{\partial u^n}{\partial x} \tag{3.16}$$

can be gained by reformulation.

3.4.4 Semi-Lagrangian Implicit Runge-Kutta

The semi-Lagrangian method is combined with the implicit Runge-Kutta method to treat the diffusive term comparable to the IMEX method. This combination will be referred to as semi-Lagrangian Runge-Kutta method. Here, the second-order version is described. The first-order version works analogously.

For the second-order version, the second-order semi-Lagrangian method is paired with a second-order implicit Runge-Kutta method. Starting from the Lagrangian formulation of the Burgers equation (3.1) a discretization with IRK of second-order using the notation introduced in Sec. 3.2 and dropping the subscript a results in

$$\begin{aligned} k_1 &= \nu \frac{\partial^2}{\partial x^2}\left(u_*^n + \frac{\Delta t}{2} k_1\right) \\ u^{n+1} &= u_*^n + \Delta t k_1 \ . \end{aligned} \tag{3.17}$$

The implicit part (3.17) can again be formulated as a definite Helmholtz problem

$$\left(1 - \nu \frac{\Delta t}{2} \frac{\partial^2}{\partial x^2}\right) k_1 = \nu \frac{\partial^2 u_*^n}{\partial x^2} \ . \tag{3.18}$$

The calculation of the SLIRK can be done in two steps. First, the departure points are estimated and the current velocities are interpolated to those points. Second, the interpolated values are used to compute the right hand side before solving the definite Helmholtz problem with the accurate spectral solver.

It is necessary to use Strang splitting to reach the second-order accuracy for all values of the viscosity. Strang has provided in [99] a splitting method of second-order for the evaluation of partial differential equations. With this

splitting it is possible to apply differential operators, which are added in an equation, in succession. The method maintains a second-order approximation, if the discretizations of all operators are of second-order themselves. With both operators of the SLIRK method being of second-order, the Strang splitting is applied the following way. First, the departure points are estimated, then half a time step of the diffusion is applied. This is followed by the interpolation of the velocities to the departure points, before another half time step is run with the diffusion.

3.4.5 Discrete Adomian Decomposition Method

Finally, the discrete Adomian decomposition method (DADM) is reviewed. This method is not widely used and, therefore, the description starts with a short explanation of the Adomian decomposition method (ADM), which is the foundation of the DADM, before its discretization is discussed.

Adomian Decomposition Method

The Adomian decomposition method was developed by ADOMIAN in the 1980s and 1990s, see e.g. [7]. Its main idea is to decompose non-linearities into series of Adomian polynomials to get an exact representation of these. With the ADM it is possible to calculate the analytical solution, or, if this is not possible, a good approximation with fast convergence to the actual solution of the investigated problem. This is possible because no discretization, linearization, or perturbation theory is applied [4, 5, 7, 8].

Following the notation of [6], the ADM is applied to a deterministic equation $\mathcal{G}(u(t)) = g(t)$, where $\mathcal{G}$ is a (non-)linear operator. The operator $\mathcal{G}$ can be decomposed in $\mathcal{L}+\mathcal{R}$ and $\mathcal{N}$, where $\mathcal{L}$ is an easily invertible linear operator of highest-order, $\mathcal{R}$ are the remaining linear operators, and $\mathcal{N}$ are all non-linear operators. The equation then reads

$$\mathcal{L}u + \mathcal{R}u + \mathcal{N}(u) = g \; .$$

Let the inverse of $\mathcal{L}$ be $\mathcal{L}^{-1}$ and applying it to the equation gives

$$\mathcal{L}^{-1}\mathcal{L}u = \mathcal{L}^{-1}g - \mathcal{L}^{-1}\mathcal{R}u - \mathcal{L}^{-1}\mathcal{N}(u) \; .$$

Here, the left hand side can be evaluated to $\mathcal{L}^{-1}\mathcal{L}u = u + C$, where C are the integration constants.

At this point the idea of the ADM is used. The unknown is expanded in the series $u = \sum_{i=0}^{\infty} u_i$ and the non-linear term in the series $\mathcal{N}(u) = \sum_{i=0}^{\infty} A_i$, with the Adomian polynomials A_i. Inserting the expansions yields $u_0 = \mathcal{L}^{-1}g - C$ by comparison of the terms, which leads to

$$\sum_{i=0}^{\infty} u_i = u_0 - \mathcal{L}^{-1}\mathcal{R}\sum_{i=0}^{\infty} u_i - \mathcal{L}^{-1}\sum_{i=0}^{\infty} A_i \ .$$

With this the recursive relation

$$\begin{aligned} u_1 &= -\mathcal{L}^{-1}\mathcal{R}u_0 - \mathcal{L}^{-1}A_0 \\ u_2 &= -\mathcal{L}^{-1}\mathcal{R}u_1 - \mathcal{L}^{-1}A_1 \\ &\vdots \\ u_{i+1} &= -\mathcal{L}^{-1}\mathcal{R}u_i - \mathcal{L}^{-1}A_i \\ &\vdots \end{aligned} \tag{3.19}$$

is found.

The Adomian polynomials still have to be defined. These can formally be written as

$$A_i(u_0, u_1, \ldots, u_i) = \frac{1}{i!}\left[\frac{\mathrm{d}^i}{\mathrm{d}\lambda^i}\mathcal{N}\left(\sum_{j=0}^{\infty}\lambda^j u_j\right)\right]_{\lambda=0} , \tag{3.20}$$

see [1, 4, 5, 7, 27]. The polynomial A_i is dependent only on the unknowns $u_0, u_1, \ldots, u_i$, which according to (3.19) are already known at the time A_i being calculated. This sparks the idea of a parallel computation of the terms of the polynomial.

The expansion of the non-linear term is a generalization of the Taylor series. Here, the series is build in the neighborhood of a function, instead of a point

$$\mathcal{N}(u) = \sum_{i=0}^{\infty} A_i = \sum_{i=0}^{\infty} \frac{1}{i!}\left(u - u_0\right)^i \mathcal{N}^{(i)}\left(u_0\right) \ .$$

Discretization of the Adomian Decomposition Method

The Adomian decomposition method as described in the previous section is intended to be used analytically to calculate the exact solution. In this section, a discretized version is discussed, such that the ADM can be used numerically.

The first deviation from the ADM is the truncation of the expansion series. Such an approximation of the solution shows already with a few terms a good accuracy [9, 35, 39, 62, 63]. In [26, 40, 61] a small convergence radius was shown as the issue of such a truncation. The resulting small maximal time, which could be computed with the truncated series, was circumvented in [2] by using ADM as time stepper, still using continuous space. Zhu et al. [111] then used the ADM iteratively on a discretized domain coining the term discrete Adomian decomposition method. Their ansatz is comparable to the one described here, therefore, the name is adapted.

As mentioned, the expansion series have to be truncated. Denoting the order of approximation with p gives

$$\sum_{i=0}^{\infty} u_i \approx \sum_{i=0}^{p} u_i = u_0 - \mathcal{L}^{-1}\mathcal{R}\sum_{i=0}^{p-1} u_i - \mathcal{L}^{-1}\sum_{i=0}^{p-1} A_i \ .$$

If the approximation at the discrete time $t_n = t_0 + n\Delta t$ is defined as $\bar{u}^n$ and the initial condition as $\bar{u}^0 = u_0$, the time-stepping scheme of order p can be written as

$$\bar{u}^{n+1} = \bar{u}^n + \sum_{i=1}^{p} u_i \ .$$

One step with this time-stepping scheme requires p evaluations of the linear and $\mathcal{O}(p^2)$ evaluations of the non-linear term.

DADM Applied to Burgers' Equation

When applying the DADM to the Burgers equation the operators $\mathcal{L}u = \partial u/\partial t$, $\mathcal{R}u = -\nu\partial^2 u/\partial x^2$, and $\mathcal{N}(u) = u\partial u/\partial x$ can be identified. With these the time discrete recursion of the truncated series reads

$$\begin{aligned} u_0 &= \bar{u}^n \\ u_1 &= -\frac{\Delta t}{1}(\mathcal{R}u_0 + A_0) \\ u_2 &= -\frac{\Delta t}{2}(\mathcal{R}u_1 + A_1) \\ &\vdots \\ u_p &= -\frac{\Delta t}{(p-1)}(\mathcal{R}u_{p-1} + A_{p-1}) \,, \end{aligned} \tag{3.21}$$

where the independence of $\bar{u}^n$ on t is used, such that the integration $\mathcal{L}^{-1} = \int_{t_n}^{t_{n+1}} \cdot \, \mathrm{d}t$ is trivial. In this context the Adomian polynomials can be written as

$$A_i = \sum_{j=0}^{i} u_j \frac{\partial u_{i-j}}{\partial x} \,. \tag{3.22}$$

3.5 Parallelization

Next to the parallelization in time, which is the focus of this thesis, a parallelization in space is typically performed. Both types of parallelization are discussed in this section.

3.5.1 Parallelization in Space

The intuitive way of parallelization is the one in space. It is well-known and studied by now. Since the present work is focused on the parallelization in time, only the basic idea of parallelization in space is provided here. A good starting point for further information on domain decomposition methods is e.g. provided by Dolean et al. [36].

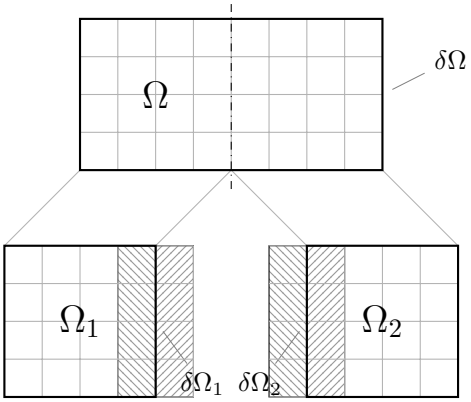


Figure 3.1: Splitting the discretized domain Ω with the boundary $\delta\Omega$ into two domains Ω_1 and Ω_2 for parallelization. The new boundaries $\delta\Omega_1$ and $\delta\Omega_2$ are given the values of the correspondingly shaded cells of the other domain

Assuming the solution in the domain Ω is to be calculated and boundary conditions are given for the boundary $\delta\Omega$. Then the domain can be split in the two domains Ω_1 and Ω_2 as shown in Fig. 3.1. Both new domains need now a boundary condition for the new boundary created by the split. For the domain Ω_1 this new boundary is $\delta\Omega_1$ and the values of the boundary are given by the values inside of Ω_2 next to the new boundary. The boundary of Ω_2 is treated respectively. Using the values of the previous iteration for the new boundaries $\delta\Omega_1$ and $\delta\Omega_2$ makes a parallel computation of the domains Ω_1 and Ω_2 possible.

Without considering the losses introduced by parallelization a cut in half of the time-to-solution is to be expected by the described method. As already hinted this is not completely true. By utilizing parallelization, three different kinds of losses are introduced which reduce the efficiency of the parallelization process. These are

1. Communication losses,
2. Numerical losses, and
3. Workload distribution losses.

Communication losses arise because of the time it takes to transfer the data between the different processing units. Numerical losses are generated by the introduction of extra boundaries, which are treated explicitly, and increase the number of arithmetic operations necessary for convergence. Workload distribution losses are present if the workload between the processing units is not balanced [88].

Especially, the communication losses can lead to a saturation of the efficiency in case of an increasing amount of processing units. An extreme case to illustrate the problem would be to consider a spatial domain, where each cell is a domain computed by one processor. In this case each processor needs the data of all surrounding processors to compute one value.

3.5.2 Parallelization in Time

The mentioned saturation limit, which is reached by the parallelization in space if the domain is split too many times, started the interest in parallel-in-time methods. Current and future hardware will provide a large amount of computing cores, which can be used. Employing all available cores of a given system might not be possible efficiently with parallel in space methods. In this case the question is, whether a parallelization in time on top can reduce the time-to-solution even further.

During the last few decades the research regarding parallel-in-time methods grew constantly. Multiple different methods were published and investigated, as Gander's review shows [47]. Those methods perform well on parabolic problems like the heat equation, but generally struggle with non-linear hyperbolic equations.

The investigation within this thesis is done with the Parareal algorithm, as it is a simple, non-intrusive algorithm, which makes it very appealing to employ it with legacy codes.

Parareal Algorithm

The Parareal algorithm was introduced by Lions et al. [71]. It can be applied when looking for a solution to an autonomous system. For simplicity, an ordinary differential equation

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{u} = f(\mathbf{u}), \qquad \text{with } \mathbf{u}(t_0) = \mathbf{u}_0 \tag{3.23}$$

is considered, where $f(\mathbf{u})$ is a Lipschitz continuous function.

The idea of Parareal is to split the time domain $[t_0, T]$ in N_T slices of size ΔT and compute the solutions on each of these in parallel. This means an initial condition is necessary for each time-slice $[t_n, t_{n+1}]$, where $t_{n+1} = t_n + \Delta T$. The initial conditions have to be the solutions of the previous time-slices. These solutions are not available if a parallel computation of the time-slices is desired. Therefore, the initial conditions have to be estimated. In case of Parareal, this is done by using two solvers, which can, but do not have to, be the same. One of the solvers is the fine time-stepping method using time step sizes of Δt and the other one is the coarse time-stepping method typically using ΔT for the time step size.

The estimation of the initial conditions is done with the coarse propagator denoted by the functional $\mathcal{C}(\mathbf{u}^n, t_n, t_{n+1})$, whereas the calculations on the time-slices itself are run with the fine propagator $\mathcal{F}(\mathbf{u}^n, t_n, t_{n+1})$ using multiple time steps of the smaller time step size Δt to evaluate the update from time step t_n to $t_{n+1} = t_n$. An usual choice for the fine propagator is a sequential state of the art time-stepping scheme. The coarse propagator should be computationally a lot cheaper than the fine one.

Since the estimation of the initial conditions is not an accurate representation of the solution at those points, an iterative prediction/correction scheme of the two solvers is used to gain accurate results. Within this iterative scheme the fine solver can be run in parallel as designated. The resolving iteration can be written as

$$\mathbf{u}_{k+1}^{n+1} = \mathcal{C}(\mathbf{u}_{k+1}^{n}) + \mathcal{F}(\mathbf{u}_{k}^{n}) - \mathcal{C}(\mathbf{u}_{k}^{n}) \,, \tag{3.24}$$

following the notation of Baffico et al. [11], with the initial condition $\mathbf{u}_0^0$. A pseudo code representation is provided in Alg. 1.

$\mathbf{U}_0^0 \leftarrow \tilde{\mathbf{U}}_0^0 \leftarrow \mathbf{u}^0$
for $n = 1$ **to** N_T **do**
  $\tilde{\mathbf{U}}_0^n \leftarrow \mathcal{C}\left(\mathbf{U}_0^{n-1}\right)$
  $\mathbf{U}_0^n \leftarrow \tilde{\mathbf{U}}_0^n$
end
while $|\mathbf{U}_k^n - \mathbf{U}_{k-1}^n| > tol \ \exists n$ **do**
  $\mathbf{U}_k^0 \leftarrow \mathbf{u}^0$
  for $n = 1$ **to** N_T **do**
    $\hat{\mathbf{U}}_{k-1}^n \leftarrow \mathcal{F}\left(\mathbf{U}_{k-1}^{n-1}\right)$ `// Parallel step`
  end
  for $n = 1$ **to** N_T **do**
    $\tilde{\mathbf{U}}_k^n \leftarrow \mathcal{C}\left(\mathbf{U}_k^{n-1}\right)$ `// Predict`
    $\mathbf{U}_k^n \leftarrow \tilde{\mathbf{U}}_k^n + \hat{\mathbf{U}}_{k-1}^n - \tilde{\mathbf{U}}_{k-1}^n$ `// Correct`
  end
end

Algorithm 1 Pseudo code of the Parareal algorithm. $\mathcal{C}$ and $\mathcal{F}$ denote the coarse and fine solver, respectively. The initial condition is $\mathbf{u}^0$, $\mathbf{U}_k^n$ denotes the solution at iteration k and time point t_n. The solutions of $\mathcal{C}$ and $\mathcal{F}$ are $\tilde{\mathbf{U}}_k^n$ and $\hat{\mathbf{U}}_k^n$ respectively. N_T is the number of time-slices. This representation of the algorithm was originally published in [89]

Based on the accuracy and smoothness of the fine and coarse solver it is possible to estimate an analytical upper bound of the error achieved after a certain number of iterations [12, 50]. A parallelization with Parareal yields clearly no acceleration if the number of iterations reaches or exceeds the number of time-slices of the problem at hand. Actually, even matching the number of time-slices would already lead to an increased wall-clock time compared to a serial run, as additional overhead is introduced by Parareal. Two requirements have to be fulfilled to gain acceleration with Parareal:

1. The coarse propagator $\mathcal{C}$ has to be way cheaper than the fine propagator $\mathcal{F}$ per time-slice. This can be achieved with $\Delta t \ll \Delta T$ or by choosing

a computationally cheaper time-stepping scheme for the coarse solver than for the fine solver.

2. Parareal needs way fewer iterations than the number of time-slices.

Those two requirements are contradicting each other. The main challenge is to find an adequate coarse solver to maximize the speedup. This challenge is problem specific and has to be solved for each investigated problem individually.

Parareal with SLIRK

The semi-Lagrangian method described in Sec. 3.2 needs the results of time step t_n and t_{n-1} to calculate time step t_{n+1} with second-order accuracy. This poses additional requirements on the communication of the Parareal algorithm, if the SLIRK algorithm is to be used as the coarse solver. These requirements can be fulfilled by (a) additional interfaces for the communication, or (b) an increase in memory usage. This is important for a parallel implementation of the algorithm and its efficiency, so this is discussed here.

When Alg. 1 is expanded with the two steps used in the semi-Lagrangian method, the SLIRK-Parareal shown in Alg. 2 is reached. The difference to the standard algorithm is the necessity of the additional velocity $\mathbf{U}_{*k}$. This velocity can be handled differently depending on whether the used system has shared or distributed memory:

1. In case of a shared memory system or a partitioned global address space the additional velocity can be simply stored as a new variable and is globally accessible by a pointer.

2. For a distributed system it is necessary to include additional communication as described below.

The code used for this thesis is serial, however, its implementation resembles the communication on distributed memory systems, as parallel-in-time algorithms target large systems. On a distributed memory system each time-slice has its independent memory areas. For the standard Parareal algorithm the

$\mathbf{U}_0^0 \leftarrow \underline{\mathbf{U}_0^{-1}} \leftarrow \tilde{\mathbf{U}}_0^0 \leftarrow \mathbf{u}^0$
for $n = 1$ **to** N_T **do**
  $\underline{\mathbf{U}_{*0} = \mathcal{SLIRK}\left(\mathbf{U}_0^{n-1}, \mathbf{U}_0^{n-2}\right)}$
  $\tilde{\mathbf{U}}_0^n \leftarrow \mathcal{C}\left(\underline{\mathbf{U}_{*0}}\right)$
  $\mathbf{U}_0^n \leftarrow \tilde{\mathbf{U}}_0^n$
end
while $|\mathbf{U}_k^n - \mathbf{U}_{k-1}^n| > tol \ \exists n$ **do**
  $\mathbf{U}_k^0 \leftarrow \underline{\mathbf{U}_k^{-1}} \leftarrow \mathbf{u}^0$
  for $n = 1$ **to** N_T **do**
    $\hat{\mathbf{U}}_{k-1}^n \leftarrow \mathcal{F}\left(\mathbf{U}_{k-1}^{n-1}\right)$ `// Parallel step`
  end
  for $n = 1$ **to** N_T **do**
    $\underline{\mathbf{U}_{*k} = \mathcal{SLIRK}\left(\mathbf{U}_k^{n-1}, \mathbf{U}_k^{n-2}\right)}$
    $\tilde{\mathbf{U}}_k^n \leftarrow \mathcal{C}\left(\underline{\mathbf{U}_{*k}}\right)$ `// Predict`
    $\mathbf{U}_k^n \leftarrow \tilde{\mathbf{U}}_k^n + \hat{\mathbf{U}}_{k-1}^n - \tilde{\mathbf{U}}_{k-1}^n$ `// Correct`
  end
end

Algorithm 2 Pseudo code of the SLIRK-Parareal algorithm, where the semi-Lagrangian formulation with SETTLS is used as the coarse solver. Parts added to the standard algorithm are underlined. $\mathbf{U}$ denotes the solution of the algorithm, $\tilde{\mathbf{U}}$ and $\hat{\mathbf{U}}$ the solution of the coarse and fine solver, respectively. The superscript n stands for the time step, the index k for the Parareal iteration and the index $*$ for the evaluation at the departure point. N_T is the number of time-slices. This algorithm was originally published in [89]

result of the time-slice $T_{n-1} = [t_{n-2}, t_{n-1}]$ is send as the initial condition to the time-slice $T_n = [t_{n-1}, t_n]$. The complete initial condition for time-slice T_n with the SLIRK-Parareal algorithm needs the result of time-slice $T_{n-2} = [t_{n-3}, t_{n-2}]$ as well.

This makes a send from T_{n-2} necessary, as $\mathbf{U}_k^{n-1}$, which contained the data before, is overwritten with $\mathbf{U}_{*k}$ in time-slice T_{n-1} to reduce the memory consumption. A sketch of the communication within the SLIRK-Parareal is

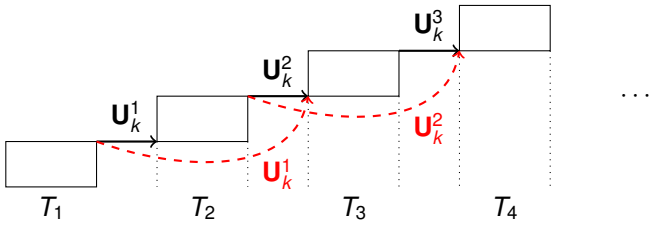


Figure 3.2: Sketch of the communication pattern for the prediction and correction step (as boxes) of the distributed memory Parareal algorithm in iterations k with the semi-Lagrangian formulation applied to the coarse solver. The time-slices are denoted by $T_n = [t_{n-1}, t_n]$. Arrows show the communication of the velocity $\mathbf{U}_k^n$ (solid: standard Parareal; dashed: additional for SLIRK-Parareal). This sketch was originally published in [89]

provided in Figure 3.2. Here, the solid arrows denote the standard communication and the dashed arrows indicate the additional communication necessary for the SLIRK-Parareal.

3.6 Method of Manufactured Solutions

The Method of Manufactured Solutions (MMS) is a method used to verify an implementation by generating an analytical solution for the code. This method is purely mathematical and can therefore be used for all kinds of applications [83].

In short, this method is used to create an exact solution to test against without considering the physical correctness of this solution. This can be achieved by introducing a source term $Q(t, x)$ to the equation at hand. Here, for simplicity the equation is one dimensional, but the method works the same in higher dimensions. This source is then used to enforce the exact solution.

In case of Burgers' equation, (2.12) with the additional source term reads

$$\frac{\partial u(t,x)}{\partial t} + u(t,x)\frac{\partial u(t,x)}{\partial x} = \nu \frac{\partial^2 u(t,x)}{\partial x^2} + Q(t,x)\ . \tag{3.25}$$

Now an exact solution can be chosen. The exact solutions used with this method should be closed form functions, which are differentiable at least up to the highest-order of the differential operators in the equation at hand. For example, a solution could be constructed with trigonometric and polynomial functions. Guidelines for such constructions are provided in [87].

For this example, the solution $\hat{u}(t, x) = \sin(t + x)$ is used. The source term Q to force the exact solution is calculated by inserting $\hat{u}$ into (3.25), which yields

$$Q(t, x) = \cos(t + x) + \sin(t + x)\cos(t + x) + \nu \sin(t + x) \; . \tag{3.26}$$

Initial and boundary conditions are evaluated from $\hat{u}$. This source term is then implemented in the code to recalculate the numerical approximation $\bar{u}$ of the exact solution $\hat{u}$.

The verification of the implementation is then possible on one hand by directly comparing $\bar{u}$ and $\hat{u}$. On the other hand, by refining the spatial or temporal discretization the order of the numerical scheme used for the discretization can be verified.

4 Verification and Details of the Implementation

All the discussed numerical methods, which are used in the following chapters were implemented. The implementation was carried out within the SWEET framework [92], which already supplies a verified implementation of the described spatial discretization. The chosen spatial discretization limits the boundary conditions to periodic boundary conditions, which are used for all calculations throughout this thesis. An advantage of the SWEET framework is the simple inclusion of new time-stepping schemes, which made it appealing for this thesis.

Additional information necessary for the implementation of the numerical methods is given in this chapter. Furthermore, a verification of the implementation is carried out.

The verification is done by comparing the numerically and analytically calculated convergence order of the implemented methods. The difference between the numerical solution and the analytical solution given by the Cole-Hopf transformation yields the discretization error. By successively decreasing the discretization width, it is possible to compute the numerical convergence order from the discretization error. If the analytical and the numerical convergence order do not match, an error in the implementation is to be suspected.

All verifications are run with the same benchmark. This benchmark consists of a sinusoidal wave as an initial condition. This initial condition reads

$$u_0(x) = \sin(2\pi x)$$

and snapshots of the corresponding time evolution are shown in Fig. 4.1 for the space-time domain $(x, t) \in [0, 1]^2$. Here, the viscosity is set to $\nu = 0.01$ such that both the advective and the diffusive term have an influence on

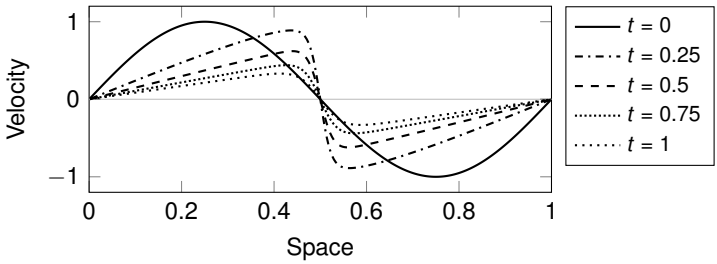


Figure 4.1: Solution of the Burgers equation with $\nu = 0.01$ to a sinus wave initial condition over the spatial domain at every 0.25 time units in the time interval $t \in [0, 1]$

the solution. For the discretization a grid spacing of $\Delta x = 1/256$ equating $M = 170$ spectral modes is used. The time step widths are taken from the set $\Delta t \in \{1.25 \times 10^{-4}, 2.5 \times 10^{-4}, 5 \times 10^{-4}\}$.

The numerical results calculated based on this benchmark $\bar{u}$ are compared to the analytical solution $\hat{u}$ to calculate the discretization error ε. The errors of two different discretizations denoted by h and $2h$ can then be used to calculate the convergence order

$$p = \frac{\log\left(\frac{\varepsilon_{2h}}{\varepsilon_h}\right)}{\log\left(\frac{2h}{h}\right)}\,, \tag{4.1}$$

see e.g. [88]. As the notation suggests typically a factor of two lies between the discretization widths, as it is also the case with the time step widths chosen for this benchmark.

Within this verification, the discretization error is the maximal absolute difference between the numerical and analytical solution $\varepsilon = \max(|\bar{u} - \hat{u}|)$ at the final time $T = 1$.

4.1 Spectral Method

The transformation between the physical and spectral space is carried out with the fast Fourier transform (FFT) algorithm. In its discrete form, the discrete fast Fourier transform (DFFT) algorithm, the transformation is carried out using an even number of Fourier modes. These modes are $m \in \{-M/2 + 1, -M/2 + 2, \ldots, M/2\}$, where M is the total number of Fourier modes. A representation with an odd number of modes ($m \in \{-M/2, \ldots, M/2\}$) would be notation-wise closer to the continuous Fourier series, but a DFFT with an odd number of modes is less efficient.

A 2/3 anti-aliasing technique is employed, as mentioned in Sec. 3.3. The effect of this technique leads to the resolution in the physical space being 1.5-times higher, than the one in spectral space. This is the reason for the difference in physical ($N = 256$) and spectral ($M = 170$) resolution noted above. The technique also ensures an even resolution in physical space.

Finally, it has to be mentioned, that only half of the spectral modes are computed with this implementation. This is possible, since the modes $m \in \{-M/2 + 1, \ldots, -1\}$ are redundant for solutions which are only real valued in physical space.

4.2 Cole-Hopf Transformation

The Cole-Hopf transformation is implemented to gain the analytical results of the benchmark for comparison with the numerical results of the other time-stepping schemes.

To gain the solution, the equations (2.16), (2.15), and (2.14) have to be solved. Due to the chosen spatial discretization with the Fourier basis, solving derivatives converts to a simple multiplication. Therefore, equations (2.15), and (2.14) are solved easily. The integral in (2.16), however, imposes a restriction on the initial condition u_0. Although an integration can be carried out as a simple division in spectral space, the integration of a constant value can not be represented in a discrete spectral space. Because of this only initial conditions whose zeroth mode is equal to zero can be calculated with the

implementation done for this thesis. This is in parts the reason for the chosen benchmark.

The verification of the implemented method is done following the suggestion of Wood [109], who introduced the exact solution

$$\hat{u} = \frac{2\nu\pi \exp(-\pi^2 \nu t) \sin(\pi x)}{a + \exp(-\pi^2 \nu t) \cos(\pi x)}, \qquad \text{with } a > 1$$

to the viscous Burgers equation with the initial condition

$$u_0 = \frac{2\nu\pi \sin(\pi x)}{a + \cos(\pi x)}, \qquad \text{with } a > 1 \, .$$

With a slight modification it is possible to scale the initial condition

$$u_0 = \frac{4\nu\pi \sin(2\pi x)}{a + \cos(2\pi x)}, \qquad \text{with } a > 1 \, .$$

and with this also the analytical solution

$$\hat{u} = \frac{4\nu\pi \exp(-4\pi^2 \nu t) \sin(2\pi x)}{a + \exp(-4\pi^2 \nu t) \cos(2\pi x)}, \qquad \text{with } a > 1$$

in a way, such that a full period of the function lies inside the investigated spatial domain $0 \leq x \leq 1$. The comparison between numerical and analytical solution is carried out at $t = 1$ with $\nu = 0.01$ and $a = 2$ over the given spatial domain.

The maximal difference between the analytical solution and the one computed with the SWEET framework over all spatial points is $\varepsilon = 7.67 \times 10^{-16}$. This error is in the range of machine precision. The implementation of the Cole-Hopf transformation is thus verified.

4.3 Explicit Runge-Kutta

The first time-stepping scheme to be verified is the explicit Runge-Kutta method. The implementation is done for the orders $p \in \{1, \ldots, 4\}$. For all these orders the most basic schemes are used.

Running the benchmark scenario with all four orders of the explicit Runge-Kutta method leads to the results shown in Table 4.1. Reducing the time

Table 4.1: Validation of the explicit Runge-Kutta method implementation by comparing the numerical and theoretical order of convergence. The theoretical p order is given by the notation ERKp

Scheme	Time Step Size	Max. Abs. Error	Calculated Order
ERK1	5×10^{-4}	$3.25216912828 \times 10^{-4}$	
	2.5×10^{-4}	$1.62532282331 \times 10^{-4}$	1.0007
	1.25×10^{-4}	$8.12471081713 \times 10^{-5}$	1.0003
ERK2	5×10^{-4}	$1.97971810686 \times 10^{-7}$	
	2.5×10^{-4}	$4.94959064087 \times 10^{-8}$	1.9999
	1.25×10^{-4}	$1.23744926887 \times 10^{-8}$	1.9999
ERK3	5×10^{-4}	$7.73428231576 \times 10^{-11}$	
	2.5×10^{-4}	$9.67137262278 \times 10^{-12}$	2.9995
	1.25×10^{-4}	$1.18813945508 \times 10^{-12}$	3.0250
ERK4	1×10^{-3}	$3.31878969498 \times 10^{-12}$	
	5×10^{-4}	$2.24065001805 \times 10^{-13}$	3.8887
	2.5×10^{-4}	$7.79715883229 \times 10^{-14}$	1.5229

step size by a factor two leads to a reduction in the discretization error by a factor of two for the first-order method. This is the expected behavior of convergence order one. In case of the second-order ERK method, the calculated numerical convergence order is also second-order implying a correct implementation. The results of the calculated numerical convergence order of $p = 2.9995$ and $p = 3.0250$ also verify the third-order method. The difference to the expected order of three is in both cases less than five percent. The fourth-order method, however, shows a numerical order of $p = 1.5229$ for the first discretization refinement of the benchmark. An additional run with $\Delta t = 1 \times 10^{-3}$ reveals the mismatch of the analytical and numerical order being due to the spatial discretization error dominating the total error, since the calculated convergence order between $\Delta t = 1 \times 10^{-3}$ and $\Delta t = 5 \times 10^{-4}$ is $p = 3.8887$. This again is only a deviation below five percent of the expected

Table 4.2: Validation of the implicit-explicit Runge-Kutta method implementation by comparing the numerical and theoretical order of convergence. The theoretical p order is given by the notation IMEXp

Scheme	Time Step Size	Max. Abs. Error	Calculated Order
IMEX1	5×10^{-4}	$2.41430105308 \times 10^{-4}$	
	2.5×10^{-4}	$1.20627448547 \times 10^{-4}$	1.0001
	1.25×10^{-4}	$6.02918456371 \times 10^{-5}$	1.0001
IMEX2	5×10^{-4}	$1.67677855937 \times 10^{-7}$	
	2.5×10^{-4}	$4.19304229332 \times 10^{-8}$	1.9996
	1.25×10^{-4}	$1.04840856220 \times 10^{-8}$	1.9998

order, which verifies the fourth-order method. With this all four methods are verified and a correct implementation is shown.

In addition, the result of the fourth-order computation shows the high accuracy of the spatial discretization. The error is first bound by the error of the spatial discretization for values of the order $\mathcal{O}(10^{-13})$. Therefore, it can be concluded that the spatial error is not much larger than the machine precision.

4.4 Implicit-Explicit Runge-Kutta

In case of the implicit-explicit Runge-Kutta method two different combinations of implicit and explicit Runge-Kutta schemes were chosen to build one first and one second-order method. The first-order method combines the explicit and implicit Euler method. For the second-order a combination of the explicit and implicit midpoint rule.

The results gained by running the benchmark are listed in Table 4.2. Here, the table shows a numerical convergence order of one for the refinement in the time step width for IMEX1. The IMEX2 method produces a convergence order of two. Hence, the implementation of the two IMEX schemes is verified.

4.5 Semi-Lagrangian Runge-Kutta

The semi-Lagrangian Runge-Kutta method is implemented in first and second-order. Regarding the second-order implementation, a short remark has to be made. The stopping criterion for the iteration (3.7) is the maximum of the absolute distance between the departure point of two consecutive iterations. A difference threshold of $tol = 1 \times 10^{-8}$ is used in combination with a fixed stopping criterion of maximal 10 iterations. Generally only a few iterations are necessary to obtain very accurate departure points.

For the calculation of the convergence order of the semi-Lagrangian Runge-Kutta method, the benchmark has to be adapted. In contrast to the other implemented methods, the error of the SLIRK method is dependent on the physical spatial discretization. Therefore, it is necessary to reduce the grid spacing to $\Delta x = 1/512$, to see the discretization error in time. Finding a good combination of spatial and temporal discretization to show the convergence order is not simple due to the multiple influences on the error, as shown in [77]. In addition, the final time of the calculation is set to $T = 1 \times 10^{-3}$, such that the influence of the error due to no mass conservation is as small as possible. With these settings, it is not possible to calculate the analytical solution with the implementation of the Cole-Hopf transformation. The values calculated with (2.16) become so large, that during the calculation a loss of significance occurs. Therefore, the maximal absolute velocity of the result is used to calculate the order. This can be achieved by

$$p = \frac{\log\left(\dfrac{u_{2h} - u_{4h}}{u_h - u_{2h}}\right)}{\log\left(\dfrac{2h}{h}\right)} ,$$

where u_h, u_{2h}, and u_{4h} denote the solutions for the discretizations h, $2h$, and $4h$ [88]. Here, a constant factor between the discretizations is necessary. Since three solutions are necessary to calculate the order, the solution to the time step size $\Delta t = 1 \times 10^{-3}$ is calculated in addition to the ones given by the benchmark.

The convergence orders computed can be found in Table 4.3. For the im-

Table 4.3: Validation of the semi-Lagrangian Runge-Kutta method implementation by comparing the numerical and theoretical order of convergence. The theoretical p order is given by the notation SLIRKp

Scheme	Time Step Size	Max. Abs. Velocity	Calculated Order
SLIRK1	1×10^{-3}	0.249797787818	
	5×10^{-4}	0.249800236538	
	2.5×10^{-4}	0.249801460508	1.0005
	1.25×10^{-4}	0.249802072404	1.0002
SLIRK2	1×10^{-3}	0.249800217246	
	5×10^{-4}	0.249802067536	
	2.5×10^{-4}	0.249802529881	2.0007
	1.25×10^{-4}	0.249802645543	1.9991

plementation of the first-order, a very good match between the analytical and numerical order is found. The results gained with the second-order method also show a matching convergence order as well. With these results, the implementation of the semi-Lagrangian Runge-Kutta is verified.

4.6 Discrete Adomian Decomposition Method

The DADM has been implemented by calculating the recursion (3.21) and the polynomials (3.22). Even with the simple benchmark given above, it becomes obvious, why a discretization in space is necessary for the DADM. Because of the discretization in time, a small convergence radius is expected. So it is necessary to calculate multiple steps to reach the final time. Each step needs an initial value, which is the solution of the previous step. Since already the initial value of the second step is more complex than the sinusoidal wave, the differentiations needed in space during the evaluation become more expensive.

It is possible to choose an arbitrary order for the DADM, but the increasing order is achieved by the recursion (3.21). The orders $p \in \{1, 2, 3, 4\}$ are tested

Table 4.4: Validation of the discrete Adomian decomposition method implementation by comparing the numerical and theoretical order of convergence. The theoretical p order is given by the notation DADMp

Scheme	Time Step Size	Max. Abs. Error	Calculated Order
DADM1	5×10^{-4}	$3.25216912809 \times 10^{-4}$	
	2.5×10^{-4}	$1.62532282414 \times 10^{-4}$	1.0007
	1.25×10^{-4}	$8.12471082296 \times 10^{-5}$	1.0003
DADM2	5×10^{-4}	$2.75543913310 \times 10^{-7}$	
	2.5×10^{-4}	$6.88681511086 \times 10^{-8}$	2.0004
	1.25×10^{-4}	$1.72148370509 \times 10^{-8}$	2.0002
DADM3	5×10^{-4}	$1.96383352978 \times 10^{-10}$	
	2.5×10^{-4}	$2.45593175189 \times 10^{-11}$	2.9993
	1.25×10^{-4}	$3.04379808540 \times 10^{-12}$	3.0123
DADM4	1×10^{-3}	$5.47074904214 \times 10^{-12}$	
	5×10^{-4}	$3.71800903193 \times 10^{-13}$	3.8791
	2.5×10^{-4}	$8.77827629631 \times 10^{-14}$	2.0825

to verify the implementation. By this the functionality of the recursion is verified and with this also the higher orders are expected to be verified.

The results gained with the benchmark are provided in Table 4.4. A decreasing error can be noted for all investigated orders when the time step size is decreased, showing the same exception as with the ERK method. The calculated numerical orders match the analytical convergence orders of the schemes. Furthermore, the errors converge to zero implying a correct representation of the Burgers equation for infinite small discretization widths.

Higher-order schemes reach already errors in the range of machine precision. Therefore, it is not possible to calculate the order based on those errors. Nevertheless, the results for the orders $p \in \{1, 2, 3, 4\}$ already verify the implementation.

4.7 Parareal

The Parareal algorithm used for the investigations throughout this thesis is a serial implementation. The implementation represents a distributed memory system, where each time-slice has its independent memory space. Furthermore, the implementation simulates one processor per time-slice used.

In the used implementation convergence of the algorithm is reached, if the difference between the maximal absolute velocity of the previous and current iteration at the final time of all time-slices is below a given tolerance tol. This threshold is an input parameter for the calculation.

The verification of the Parareal implementation is done with two distinct test cases. Both have the initial condition of the given benchmark being used in common. In addition, the fine solver for both tests is the fourth-order ERK method with a discretization width of $\Delta t = 1.25 \times 10^{-4}$. The convergence threshold is set to $tol = 1 \times 10^{-6}$. The test cases are:

1. This first test uses an arbitrary function of $u(t) = 10 * t$ which does not represent the correct solution as the coarse time stepper with 50 time-slices. The goal is to show the convergence after K iterations, where K is equal to the number of time-slices (processors) P. The property of Parareal of convergence in a maximum of K iterations was proven by Gander & Vandewalle [53].

2. The second test uses the fine time-stepping method as the coarse one as well to illustrate immediate convergence. To reduce the number of time-slices, a final time of $T = 5 \times 10^{-2}$ is used.

Setting the coarse solver to a solution, which is far away from the correct solution of the problem, leads to a convergence after K iterations. This is found with the first test. Here, the maximal difference between the current and previous iteration is at least $\varepsilon = 0.98$ for all iterations. After the K-th iteration, in this case the 50-th, the solution of the fine solver is matched up to machine precision.

In case of choosing both fine and coarse propagator the same, convergence is reached with the second iteration. This is the expected fastest convergence

possible, since for convergence with the given implementation a difference between two iterations has to be carried out. The maximal difference between the two iterations is $\varepsilon = 4.62 \times 10^{-15}$. The solution was already found in the first iteration, as the difference is in the range of machine precision.

The combination of both convergence up to machine precision after a maximum of K iterations for a badly chosen coarse solver and an immediate convergence to machine precision for a accurate coarse solver verifies the correct implementation of the Parareal algorithm.

5 Degrees of Parallelism within DADM

In this chapter, the possibility of a parallelization within the DADM method is examined. Methods like the Richardson extrapolation [81, 82] and RIDC [28] show a small scale parallelization potential for time-stepping schemes. Here, a look is taken at the degrees of parallelism of the discrete Adomian decomposition method.

In addition to analyzing the possible degrees of parallelism, it is investigated whether the DADM is a viable option as a time-stepping method and whether exploiting the parallelism is beneficial. The viability is checked by comparing the DADM to the established explicit Runge-Kutta method. This comparison is carried out both analytically and numerically. A comparison between the Adomian decomposition method and the ERK method was already performed [55, 94, 104], however, the comparisons were done neither with the Burgers equation, nor the discrete version of the ADM. The conclusions were that using the same number of terms of the truncated ADM as the order of the ERK yields results of similar accuracy. Applied to the Lorenz equation the ADM allows larger time steps, as it was shown in [55].

Parts of the content of this chapter have been published by the author et al. in [90].

5.1 Comparison expl. ERK with DADM

The first step to investigate whether the discrete Adomian decomposition method is a viable time-stepping scheme is to compare it to a similar already established scheme. For the comparison the explicit Runge-Kutta method was chosen. This choice is based on the fact that both methods can be formulated in different orders of accuracy. In addition, both methods are explicit methods which have a limited maximal stable time step size.

For better readability, two notations are introduced. First, the DADM of order p is denoted as DADMp and equally the notation ERKp is used. Second, the notation of the non-linearity of the Burgers equation identified in Sec. 3.4.5 is generalized to $\mathcal{N}(f(u), g(u)) = f(u)\partial g(u)/\partial x$.

5.1.1 First Order

For the first-order the DADM1 is compared with the explicit Euler method, which is ERK1. The DADM1, using the notation of Sec. 3.4.5, consists only of the first Adomian polynomial and reads

$$\begin{aligned}\bar{u}^{n+1} &= \bar{u}^n + \sum_{i=1}^{1} u_i \\ &= \bar{u}^n - \Delta t \left(\mathcal{R}\bar{u}^n + \mathcal{N}\left(\bar{u}^n, \bar{u}^n\right)\right) \ .\end{aligned}$$

By using the same notation for ERK1

$$u^{n+1} = u^n - \Delta t \left(\mathcal{R}u^n + \mathcal{N}\left(u^n, u^n\right)\right) \ ,$$

it can be seen that both methods are equal.

5.1.2 Second Order

Rewriting DADM2

$$\begin{aligned} u_1 &= -\Delta t \left(\mathcal{R}\bar{u}^n + \mathcal{N}\left(\bar{u}^n, \bar{u}^n\right)\right) \\ u_2 &= -\Delta t \left(\mathcal{R}u_1 + \mathcal{N}\left(u_1, u_1\right)\right) \\ \bar{u}^{n+1} &= \bar{u}^n + \sum_{i=1}^{2} u_i \end{aligned}$$

in a way dependent only on $\bar{u}^n$ leads to

$$\begin{aligned}\bar{u}^{n+1} &= \bar{u}^n - \Delta t \left(\mathcal{R}\bar{u}^n + \mathcal{N}\left(\bar{u}^n, \bar{u}^n\right)\right) \\ &\quad + \frac{\Delta t^2}{2}\Big(\mathcal{R}^2\bar{u}^n + \mathcal{R}\mathcal{N}\left(\bar{u}^n, \bar{u}^n\right) + \mathcal{N}\left(\mathcal{R}\bar{u}^n + \mathcal{N}\left(\bar{u}^n, \bar{u}^n\right), \bar{u}^n\right) \\ &\quad + \mathcal{N}\left(\bar{u}^n, \mathcal{R}\bar{u}^n + \mathcal{N}\left(\bar{u}^n, \bar{u}^n\right)\right)\Big) \ . \end{aligned} \tag{5.1}$$

Equation (5.1) is the exact representation of the generalized Taylor series up to second-order. This is expected as shown in Sec. 3.4.5. For the comparison, the intermediate stage of the midpoint rule (ERK2)

$$\begin{aligned} u_1 &= u^n - \Delta t \left(\mathcal{R}u^n + \mathcal{N}\left(u^n, u^n\right)\right) \\ u^{n+1} &= u^n - \Delta t \left(\mathcal{R}u_1 + \mathcal{N}\left(u_1, u_1\right)\right) \end{aligned}$$

is eliminated to get

$$\begin{aligned} u^{n+1} = {} & u^n - \Delta t \left(\mathcal{R}u^n + \mathcal{N}\left(u^n, u^n\right)\right) \\ & + \frac{\Delta t^2}{2} \left(\mathcal{R}^2 u^n + \mathcal{R}\mathcal{N}\left(u^n, u^n\right) + \mathcal{N}\left(\mathcal{R}u^n + \mathcal{N}\left(u^n, u^n\right), u^n\right)\right. \\ & \left. + \mathcal{N}\left(u^n, \mathcal{R}u^n + \mathcal{N}\left(u^n, u^n\right)\right)\right) \\ & + \frac{\Delta t^3}{4} \left(\mathcal{N}\left(\mathcal{N}\left(u^n\right), \mathcal{R}u^n\right) + \mathcal{N}\left(\mathcal{R}u^n, \mathcal{N}\left(u^n, u^n\right)\right)\right. \\ & \left. + \mathcal{N}\left(\mathcal{N}\left(u^n\right), \mathcal{N}\left(u^n\right)\right) + \mathcal{N}\left(\mathcal{R}u^n, \mathcal{R}u^n\right)\right) \ . \end{aligned} \tag{5.2}$$

A comparison between (5.1) and (5.2) shows all terms of DADM2 being present in ERK2. Besides these, there are additional terms of third-order. The additional terms are not the terms of the generalized Taylor series and, therefore, should have no impact on the convergence order of the scheme, but may have an impact on the error. This is further investigated in Sec. 5.3.3.

5.1.3 Higher Order

Expanding this comparison to higher orders shows results similar to those of the second-order. For order p DADMp is an exact representation of the generalized Taylor series cut off at the specific order. ERKp on the contrary always has additional terms of orders larger than p. In Sec. 5.3.2 it is investigated whether those additional terms have an impact on the maximal stable time step size.

5.2 Degrees of Parallelism

Considering the comparison between DADM and ERK it seems to be questionable whether DADM is a viable time-stepping method. Both schemes are

Table 5.1: The distribution of the workload on a multi-core system is shown for the parallelization of the DADM. The columns indicate from left to right the necessary communication, the data calculated on processor P_i, and the dependencies and calculations of the data. The idea of the parallelization is demonstrated for the first three velocities of the DADM. The parallelizable part is the evaluation of the non-linearities

Comm.	P_0	P_1	P_2	Calculations & dependencies
Bcast	u_0			
	A_0	A_0	A_0	$A_0 = (u_0 \cdot \nabla)u_0$
	u_1	u_1	u_1	$u_1 = -\Delta t(\nu\nabla^2 u_0 + A_0)$
Reduce	$u_0\nabla \cdot u_1$	$u_1\nabla \cdot u_0$		
	A_1	A_1	A_1	$A_1 = \sum_{i=0}^{1}(u_i \cdot \nabla)u_{1-i}$
	u_2	u_2	u_2	$u_2 = -\Delta t/2(\nu\nabla^2 u_1 + A_1)$
Reduce	$u_0\nabla \cdot u_2$	$u_1\nabla \cdot u_1$	$u_2\nabla \cdot u_0$	
	A_2	A_2	A_2	$A_2 = \sum_{i=0}^{2}(u_i \cdot \nabla)u_{2-i}$
	u_3	u_3	u_3	$u_3 = -\Delta t/3(\nu\nabla^2 u_2 + A_2)$

very similar although they need different amounts of function evaluations to compute the solution. The ERK method needs as many function evaluations or more as the order of the scheme, see 3.4.1. For the DADMp p function evaluations are necessary plus a larger number of evaluations of the non-linear term arising from the calculation of the Adomian polynomials (3.20). Because of these additional non-linear evaluations DADM is expected to be computationally more expensive than ERK.

By taking a look at the calculation of the Adomian polynomials (3.20), the data dependencies show a possibility for parallelism. To the best of the authors knowledge nobody studied this possibility before the publication in [90]. The exploitation of the additional degrees of parallelism is shown in Table 5.1 for a multi-core system. Here, the workload distribution can be seen, where the individual terms of the Adomian polynomials are com-

puted in parallel, while the polynomial itself and the next velocity update are evaluated serially on each processor. The strictly sequential evaluation of DADMp has, as stated before, a runtime complexity of $\mathcal{O}(p)$ function evaluations and $\sum_{i=1}^{p} i = \frac{1}{2}p(p+1) = \mathcal{O}(p^2)$ additional evaluations of the non-linear term. Assuming the computation is more expensive than the reduction over the sum necessary with the parallelization, the runtime complexity can be reduced to $\mathcal{O}(p)$ without the additional non-linear evaluations by exploiting the additional degrees of parallelism.

Furthermore, it is possible to evaluate some terms of the sum within the Adomian polynomials already in an earlier stage. For example, the term $u_1 \nabla \cdot u_1$ in Table 5.1 belonging to A_2 can already be computed at the time A_1 is computed. By formulating the problem as a directed acyclic graph scheduling problem it might be possible to improve the wall-clock time and the required resources.

5.3 Numerical Comparison of DADM and ERK

The comparison started in Sec. 5.1, is continued here by comparing the DADM and the ERK method to each other numerically. The focus lies on investigating the assumptions made about a comparable time step size and accuracy with slightly different error behavior between the two methods.

5.3.1 Benchmark

For this numerical comparison the following benchmark is used. This benchmark has an one dimensional Gaussian bump as the initial condition. It is given by

$$u_0(x) = e^{-50(x-0.5)^2}$$

and snapshots of the corresponding solution are shown in Fig. 5.1. The computational domain of the benchmark is $(x,t) \in [0,1]^2$. For the viscosity $\nu = 0.01$ was chosen, such that both advection and diffusion have a significant influence on the solution, as can be seen in Figure 5.1. Regarding the discretization a grid spacing of $\Delta x = 1/256$, which corresponds to

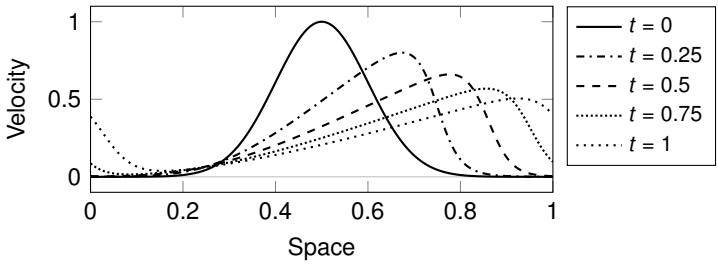


Figure 5.1: Solution of the Burgers equation with $\nu = 0.01$ to a sinusoidal wave initial condition over the spatial domain at every 0.25 time units in the time interval $t \in [0, 1]$

$M = 170$ spectral modes due to the anti-aliasing, and time step widths of $\Delta t \in \{1.25 \times 10^{-4}, 2.5 \times 10^{-4}, 5 \times 10^{-4}\}$ in case of the comparison of the convergence order are used.

Errors are calculated as the maximal absolute difference between the numerical solution and a reference solution. This reference solution is computed with a ERK method of fourth-order with a time step width of $\Delta t = 1 \times 10^{-6}$.

5.3.2 Comparison of Maximal Time Step Size

One of the possibilities to reduce the time-to-solution of a calculation is to employ large time steps. Therefore, it is of interest, whether there is a difference in the maximal possible time step with the two methods. The maximal time step $\Delta t_{\max}$ is compared for multiple orders p of the schemes. A stable time step Δt is assumed in case of the calculation of the benchmark being close to the reference solution, i.e. a maximal deviation of 10% to the reference solution was detected at the final time $t = 1$. For the ERK method the orders $p \in \{1, \ldots, 4\}$ and for the DADM the orders $p \in \{1, \ldots, 15\}$ are investigated. The maximal time step $\Delta t_{\max}$ is determined up to three significant digits.

The results are shown in Fig. 5.2. The upper two curves show the result of the benchmark. Both the ERK method and the DADM show the same results for the orders $p \in \{1, \ldots, 4\}$. Based on the DADM it can be concluded that higher orders make larger time steps possible. Using a different assump-

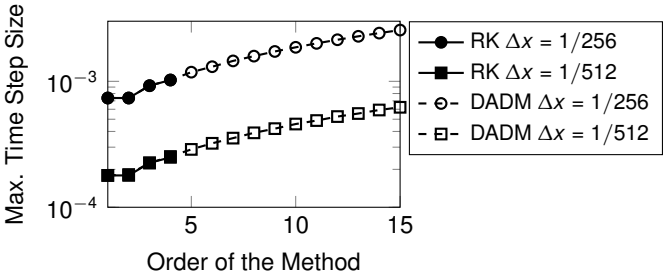


Figure 5.2: Maximal time step Δt_{max} for converged calculations in the time interval $t \in [0, 1]$ for ERK (solid) and DADM (dashed) plotted over the order of the method. Results are shown for the spatial discretization widths of $\Delta x \in \{1/256, 1/512\}$

tion for stable time step sizes, as done in [90], where every calculation not producing a 'NaN'-value is assumed stable, leads to a slightly different result. A comparison between the two assumptions yields the error amplification of DADM being smaller for even orders than the amplification of ERK, although the same maximal time step size is found for reasonable results.

In addition to the benchmark case, the same calculation was computed again with $M = 341$ spectral modes. This equals a refinement in the spatial discretization by a factor of two. The results of this calculation are given by the bottom two curves in Fig. 5.2. Here, a similar behavior as before can be seen. The smaller $\Delta t_{\max}$ are expected, because both methods are explicit and with that bound by the CFL condition. The reduction of factor two in space leads to a reduction of factor four in the maximal time step size. This shows $\Delta t_{\max}$ being bound by the diffusive flow.

All in all, there is no striking difference between the two methods used regarding the maximal time step width. Employing the parallelization described in Sec. 5.2 makes the DADM competitive to the ERK method.

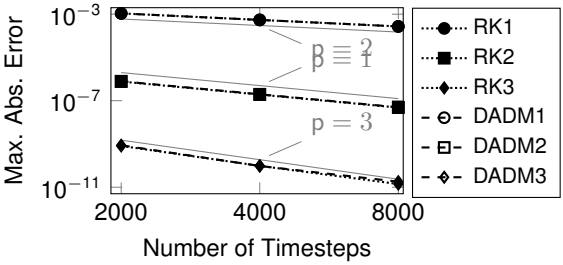


Figure 5.3: Maximal absolute error between numerical and analytic solution plotted over number of time steps to show order of convergence for ERKp and DADMp schemes of different expected orders p. In addition, the difference in error between the two methods can be seen

5.3.3 Comparison of Errors with same Convergence Order

In Sec 5.1 the possibility to formulate DADM and the ERK of the same order was stated, but the formulations are expected to have different errors due to the additional terms of the ERK method compared to the DADM. This statement is investigated in this section by a comparison of the errors for the orders $p \in \{1, 2, 3\}$.

The results are plotted in Fig. 5.3, where the maximal absolute error between the numerical solution and the reference solution at the time $t = 1$ is given for three different numbers of time steps. For the first-order methods the expected result of a first-order convergence with exactly the same error in both cases is found. This expectation is based on the fact that for first-order both methods are the explicit Euler scheme.

In case of the second-order formulations, a convergence to the second-order can be seen. For all three data points the error of the DADM is 3% higher than the error of the ERK method.

The third-order schemes show an accuracy of third-order between the time step widths of $\Delta t = 2.5 \times 10^{-4}$ and $\Delta t = 5 \times 10^{-4}$. Using (4.1) yields an order of $p = 3.177$ for DADM3 and $p = 3.069$ for ERK3. However, reducing the time step width further by a factor of two results in the orders $p = 2.415$

for DADM3 and $p = 2.793$ for ERK3. The error values themselves also show different behavior for each Δt. With $\Delta t = 5 \times 10^{-4}$ the DADM leads to an error which is 6% larger than the one of the ERK method. However, the DADM yields 1% better results with $\Delta t = 2.5 \times 10^{-4}$. As it can be seen, the finest time step leads to a larger difference between the errors of the two methods. Here, the error of DADM is 29% larger than that of the ERK method.

The fact that both schemes do not show convergence according to their order can be explained by the fact that the error for these cases lies in the range of the spatial errors. This is also the reason for the larger error with the DADM for the finest resolution. Here, the spatial error is larger than with the ERK method, since more non-linear evaluations have to be carried out, which as mentioned in Sec. 3.3 are the reason for spatial errors.

The error difference increasing with increasing order can be noted by comparing the error differences between the two methods for the coarsest time step size. Since the DADM has the larger error and the ERK method is computationally cheaper, the ERK method is the better choice for a serial calculation.

Considering the parallelization possible in the DADM it is still a viable method. For small orders the difference between the errors is insignificant. The higher errors expected by extrapolation based on the findings in Sec 5.1 can be counteracted by applying the DADM with order $p + 1$ compared to order p for the ERK method. With this the DADM is still cheaper than the ERK method considering the function evaluations which have to be done serially, if the parallelization for the DADM is used. This is due to the fact that the ERK method of order $p \geq 5$ needs $p+1$ function evaluations anyway and even more for orders of $p \geq 7$ [24].

6 Parareal with Semi-Lagrangian Formulation

In cases where the small scale parallelization investigated in the previous chapter is not sufficient, large scale parallelization methods have to be utilized. These are the parallel-in-time methods. In Section 3.5.2, as well as in the introduction, the loss of efficiency of parallel-in-time algorithms when applied to equations, which are hyperbolic, or dominantly hyperbolic, was mentioned. The reason for the loss in efficiency is in case of the Parareal algorithm its instability. In this chapter the Parareal algorithm is applied to the viscous Burgers equation with different time-stepping schemes as coarse solvers. With that it is investigated how big the influence of the stability of the coarse solver is on the stability of Parareal.

The coarse solvers tested are the ERK, IMEX, and SLIRK methods, with ERK being the least and SLIRK the most stable one. A variety of viscosities is used to handle the amount of influence of the hyperbolic term in the Burgers equation.

First studies by the author et. al. [89] revealed the ability of SLIRK to actually yield a reasonable efficiency for decreasing viscosities up to the inviscid equation. In this study time-stepping schemes of mixed-order were applied, where the advective part of the equation was discretized with second-order and the diffusive part with first-order accuracy. The investigations presented here use time-stepping schemes of either first or second-order accuracy for the whole equation.

6.1 Benchmarks

The results shown in this chapter were calculated with three different benchmarks. The first benchmark uses a Gaussian bump for the initial condition, which then is propagated over time. The other two benchmarks are based on [65]. These benchmarks use the method of manufactured solutions to have a controlled solution over the whole time frame. Kooij et al. created these test cases to model multiple length scales as can be found in turbulent flows.

All benchmarks have the spatial domain of $x \in [0, 1]$ in common. In addition, most of the Parareal settings are the same for the three benchmarks. The convergence of the Parareal algorithm is determined by the maximal absolute difference in space over all times between two consecutive iterations being below a threshold. This threshold is set to $tol = 1 \times 10^{-8}$. Furthermore, the time step of the fine solver is set to $\Delta t = 2 \times 10^{-5}$ for all Parareal studies. A total number of $N_T = 100$ time-slices is used for the calculations. Depending on the benchmark and its corresponding time step size for the coarse solver ΔT this results in multiple time steps per time-slice for the coarse solver.

The fine solver for all calculations is the second-order IMEX. The coarse solver is chosen from ERK, IMEX, and SLIRK of both first and second-order. All combinations are referred to by their respective coarse solver and its order.

6.1.1 Gaussian Bump

The first benchmark consists of a Gaussian bump initial condition, which evolves over time. The initial condition is given by

$$u(0, x) = \exp\left(-50(x - 0.5)^2\right) \ . \tag{6.1}$$

Snapshots of the solution are depicted in Fig. 6.1 for $\nu = 0.01$. The evolution of the velocity field on the spatial domain of $x \in [0, 1]$ is calculated over the time domain $t \in [0, 0.9]$. This benchmark is used both for a serial stability study and a study to compare the coarse solvers in Parareal.

For the serial stability study a combination of multiple parameter sets is used. The spatial discretization is carried out with $M \in \{42, 85, 170\}$ spectral modes equivalent to a discretization width of $\Delta x \in \{1/64, 1/128, 1/256\}$. In

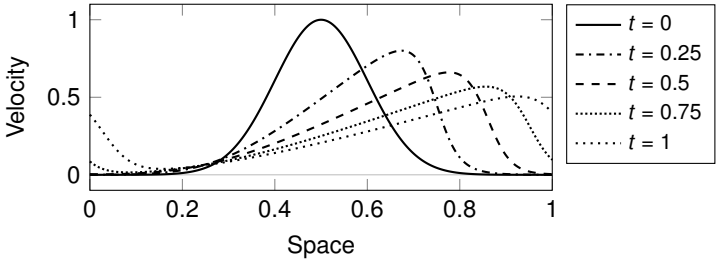


Figure 6.1: Solution of the Burgers equation with $\nu = 0.01$ to a Gaussian bump initial condition over the spatial domain $x \in [0, 1]$ at every 0.25 time units in the time interval $t \in [0, 1]$

the time domain a discretization of $\Delta t \in \{3 \times 10^{-5}, 3 \times 10^{-4}, 3 \times 10^{-3}\}$ is applied. The viscosity is chosen from $\nu \in \{0, 10^{-4}, 10^{-3}, \ldots, 1\}$.

In case of the Parareal study used to compare the different time-stepping schemes, the time step of the coarse solver is set to $\Delta T = 3 \times 10^{-3}$, which leads to three time steps per time-slice. The spatial discretization is done with $M = 170$ spectral modes, which are equivalent to a discretization width of $\Delta x = 1/256$. Only a single viscosity $\nu = 0.01$ is investigated within this study. For these settings it was not possible to calculate an analytic solution with the Cole-Hopf transformation. Therefore, the numerical solutions are compared to a solution calculated with the fourth-order ERK scheme and a time discretization width of $\Delta t = 1 \times 10^{-6}$.

6.1.2 Sinusoidal Waves

The second benchmark is the sinusoidal waves benchmark. For this benchmark the method of manufactured solutions is utilized with the solution

$$\hat{u}(t, x) = \sin(2\pi x) \sin(2\pi t) + \frac{1}{a} \sin(2\pi a x) \sin(2\pi a t) \; . \tag{6.2}$$

For this benchmark the parameter is set to $a = 3$ leading to a sum of two sinusoidal waves with one and three periods in the spacial domain. Snapshots of the solution are shown in Fig. 6.2. The space-time domain used is $(x, t) \in [0, 1]^2$. Only the first and third spectral mode have values different from

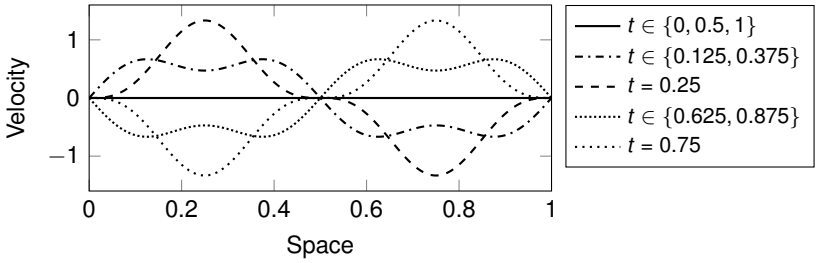


Figure 6.2: Solution of the Burgers equation with $\nu = 0.01$ to the sinusoidal waves benchmark over the spatial domain $x \in [0, 1]$ at every 0.125 time units in the time interval $t \in [0, 1]$

zero with this benchmark. Nevertheless, the spatial domain is discretized with $M = 170$ spectral modes. The time step of the coarse solver is $\Delta T = 2.5 \times 10^{-3}$, which results in four time steps per time-slice.

Since this benchmark is used to investigate the performance of Parareal with dominantly hyperbolic problems, the viscosity is chosen from the set $\nu \in \{n \times 10^{-4}, n \times 10^{-3} | n \in \{1, 2, \ldots, 10\}\}$. These viscosities already dampen the influence of the diffusion significantly.

6.1.3 Smoothened Saw-tooth Function

Comparable to the previous benchmark, the smoothened saw-tooth function benchmark is based on a known solution, which reads

$$\hat{u}(t, x) = \frac{1}{2} \sum_{a=1}^{a_{max}} \sin(2\pi a x - \pi a t + \pi a) \Psi(a, \psi) \,, \tag{6.3}$$

with

$$\Psi(a, \psi) = \frac{\psi}{\sinh(\frac{1}{2}\psi\pi a)} \,. \tag{6.4}$$

The function $\Psi(a, \psi)$ is used to reduce the influence of the larger wave numbers a, such that a smoothened saw-tooth function is achieved as it is depicted snapshot-wise in Fig. 6.3. The value $\psi = 0.1$ is used following the setting of Kooij et al. [65]. Here, the maximal a is set to $a_{max} = 85$, as this is the

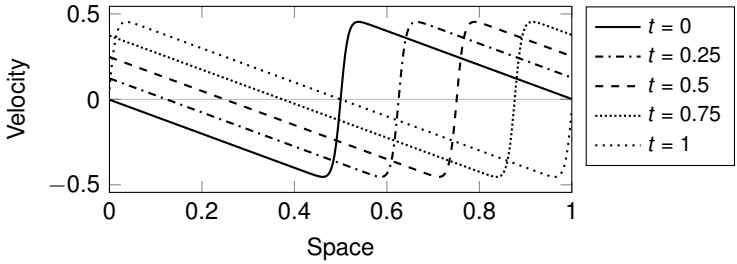


Figure 6.3: Solution of the Burgers equation with $\nu = 0.01$ to the saw-tooth benchmark over the spatial domain $x \in [0, 1]$ at every 0.25 time units in the time interval $t \in [0, 1]$

maximal number of resolvable sinus functions with $M = 170$ spectral modes. The coarse time step is set to $\Delta T = 5 \times 10^{-3}$. Hence, two steps are made per coarse time-slice.

The space-time domain and the viscosities are the same as in the previous benchmark with $(x, t) \in [0, 1]^2$ and $\nu \in \{n \times 10^{-4}, n \times 10^{-3} | n \in \{1, 2, \ldots, 10\}\}$, respectively.

6.2 Stability Study of Serial Time Stepping

The goal of the investigation carried out in this chapter is to find out whether a more stable coarse solver can improve the convergence of the Parareal algorithm. Before the solvers are applied in combination with the Parareal algorithm, their respective stability within the used parameter set is checked.

Investigating the stability of a numerical scheme can be done with an eigenanalysis. In case of non-linear ordinary differential equations ($\mathrm{d}u/\mathrm{d}t = f(u)$), like the spatially discretized Burgers equation, a linearization has to be carried out. Such a stability analysis based on the linearized equation is effective, if the Jacobian $\mathrm{d}f(u)/\mathrm{d}t$ does not change rapidly over time [69].

As the goal of this stability study is not an exact stability bound, but rather the search for a reasonable parameter combination for the following Parareal calculations, the effort is reduced to probing the parameter combinations of

Table 6.1: Results of the serial time-stepping stability study with the ERK, IMEX, and SLIRK methods of first-order for all parameter combinations of the Gaussian bump benchmark. Checkmarks indicate a stable calculation and unstable calculations are indicated with 'X'

ERK

$\Delta t = 3 \times 10^{-5}$				$\Delta t = 3 \times 10^{-4}$				$\Delta t = 3 \times 10^{-3}$			
$\nu \backslash M$	42	85	170	$\nu \backslash M$	42	85	170	$\nu \backslash M$	42	85	170
0	✓	✓	✓	0	✓	X	X	0	X	X	X
10^{-4}	✓	✓	✓	10^{-4}	✓	✓	✓	10^{-4}	X	X	X
10^{-3}	✓	✓	✓	10^{-3}	✓	✓	✓	10^{-3}	✓	✓	X
10^{-2}	✓	✓	✓	10^{-2}	✓	✓	✓	10^{-2}	✓	✓	X
10^{-1}	✓	✓	✓	10^{-1}	✓	✓	X	10^{-1}	X	X	X
1	✓	✓	X	1	X	X	X	1	X	X	X

IMEX

$\nu \backslash M$	42	85	170	$\nu \backslash M$	42	85	170	$\nu \backslash M$	42	85	170
0	✓	✓	✓	0	✓	X	X	0	X	X	X
10^{-4}	✓	✓	✓	10^{-4}	✓	✓	✓	10^{-4}	X	X	X
10^{-3}	✓	✓	✓	10^{-3}	✓	✓	✓	10^{-3}	✓	✓	✓
10^{-2}	✓	✓	✓	10^{-2}	✓	✓	✓	10^{-2}	✓	✓	✓
10^{-1}	✓	✓	✓	10^{-1}	✓	✓	✓	10^{-1}	✓	✓	✓
1	✓	✓	✓	1	✓	✓	✓	1	✓	✓	✓

SLIRK

all stable	all stable	all stable

the Gaussian bump benchmark numerically for stability. This study is important, because a stable coarse solver is necessary to gain any speedup at all with Parareal.

6.2.1 Stability of First Order Methods

The results of the calculations with the first-order schemes for all parameter combinations of the Gaussian bump benchmark are listed in Table 6.1. Parameter combinations leading to a stable computation of the benchmark are denoted by a checkmark. The ones for which the calculation diverged are denoted by an 'X'. A stable computation is assumed for cases, where the solution at the end of the time interval does not deviate more than by a factor of five from the analytical solution.

The finest time step size $\Delta t = 3 \times 10^{-5}$ leads to stable calculations for all parameter combinations of ν and M for the ERK method. The only exception is the most diffusive case with the finest spatial discretization, which yields an unstable calculation. With the medium time step size $\Delta t = 3 \times 10^{-4}$ all calculations with the viscosity $\nu = 1$ diverge. Additionally, the finest spatial discretization $M = 170$ combined with the second highest viscosity $\nu = 0.1$ leads to an unstable calculation. For the viscosity $\nu = 0$ also the two finer spatial discretizations $M \in \{85, 170\}$ lead to unstable calculations by breaking the CFL condition. Stable parameter combinations with the coarsest time step $\Delta t = 3 \times 10^{-3}$ were only achieved with the coarser spatial discretizations $M \in \{42, 85\}$ and the viscosities favoring neither diffusion, nor advection ($\nu \in \{1 \times 10^{-2}, 1 \times 10^{-3}\}$). Overall this stability pattern is expected due to the CFL condition, which has to be met with the explicit scheme.

In case of the IMEX scheme the effect of treating the diffusive term implicitly can be observed. Due to the implicit treatment stability can be achieved without fulfilling the CFL condition. All calculations with a viscosity $\nu \geq 1 \times 10^{-3}$ converge, where as all other calculations show the same stability as the ERK method. In the now stable cases the CFL condition was, therefore, broken by not resolving the diffusive information transport.

The calculations with the SLIRK scheme are stable for all parameter combinations. This is expected as the scheme is independent of the CFL condition, see Section 3.2.

6.2.2 Stability of Second Order Methods

Running the same calculations with the second-order schemes leads to the results summarized in Table 6.2. The results are overall comparable to the ones achieved with the first-order.

The second-order ERK method is more stable in the advective region than its first-order counterpart. Due to this all calculations with $\nu = 0$ for the medium time step size $\Delta t = 3 \times 10^{-4}$ are stable. With the largest time step $\Delta t = 3 \times 10^{-3}$ only the finest spatial discretization $M = 170$ leads to unstable calculations in the advective region ($\nu \in \{0, 1 \times 10^{-4}\}$). In addition, the

Table 6.2: Results of the serial time-stepping stability study with the ERK, IMEX, and SLIRK scheme of second-order for all parameter combinations of the Gaussian bump benchmark. Checkmarks indicate a stable calculation and unstable calculations are indicated with 'X'

ERK

$\nu \backslash M$	42	85	170	$\nu \backslash M$	42	85	170	$\nu \backslash M$	42	85	170
	$\Delta t = 3 \times 10^{-5}$				$\Delta t = 3 \times 10^{-4}$				$\Delta t = 3 \times 10^{-3}$		
0	✓	✓	✓	0	✓	✓	✓	0	✓	✓	X
10^{-4}	✓	✓	✓	10^{-4}	✓	✓	✓	10^{-4}	✓	✓	X
10^{-3}	✓	✓	✓	10^{-3}	✓	✓	✓	10^{-3}	✓	✓	✓
10^{-2}	✓	✓	✓	10^{-2}	✓	✓	✓	10^{-2}	✓	✓	X
10^{-1}	✓	✓	✓	10^{-1}	✓	✓	X	10^{-1}	X	X	X
1	✓	✓	X	1	X	X	X	1	X	X	X

IMEX

$\nu \backslash M$	42	85	170	$\nu \backslash M$	42	85	170	$\nu \backslash M$	42	85	170
0	✓	✓	✓	0	✓	✓	✓	0	✓	✓	X
10^{-4}	✓	✓	✓	10^{-4}	✓	✓	✓	10^{-4}	✓	✓	X
10^{-3}	✓	✓	✓	10^{-3}	✓	✓	✓	10^{-3}	✓	✓	✓
10^{-2}	✓	✓	✓	10^{-2}	✓	✓	✓	10^{-2}	✓	✓	✓
10^{-1}	✓	✓	✓	10^{-1}	✓	✓	✓	10^{-1}	✓	✓	X
1	✓	✓	✓	1	✓	✓	✓	1	✓	✓	X

SLIRK

all stable	all stable	all stable

calculation with $\nu = 1 \times 10^{-3}$ and $M = 170$ is stable for the second-order ERK method implying $\nu = 1 \times 10^{-3}$ being a viscosity setting between the advection and diffusion regions. The increased stability region of the second over the first-order ERK is according to literature [37].

As expected, the stability of the advective region ($\nu \in \{0, 1 \times 10^{-4}\}$) from the ERK method carries over to the IMEX method. In contrast to the first-order, however, the second-order IMEX shows additional unstable behavior in the diffusive region ($\nu \in \{0.1, 1\}$) with the coarsest time step $\Delta t = 3 \times 10^{-3}$ and the finest spatial discretization $M = 170$. This unintuitive result can be explained by taking a look at the formulation of the second-order IMEX, cf. Section 3.4.3. In this formulation the diffusive term is evaluated partially explicit. The partial explicit evaluation, however, still increases the stability over the fully explicit evaluation of ERK.

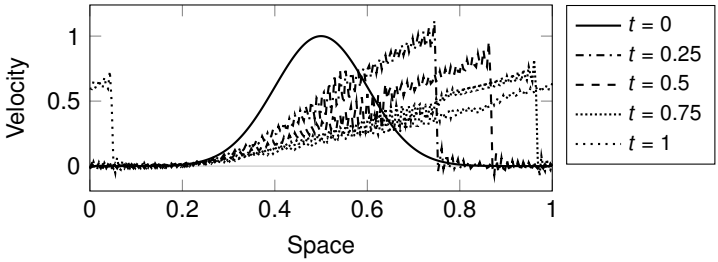


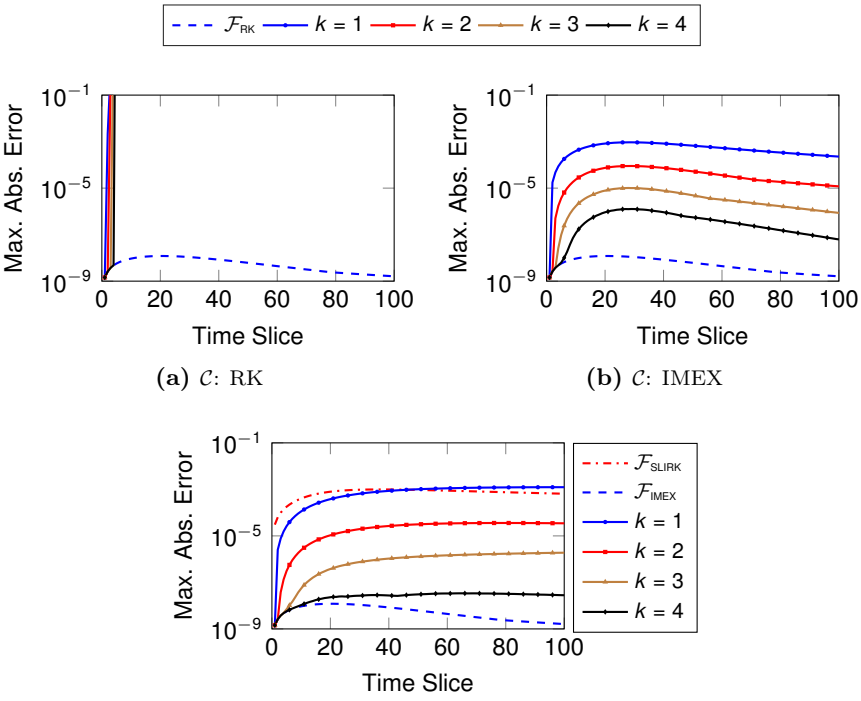
Figure 6.4: Solution of the Burgers equation with $\nu = 0.0001$ to a Gaussian bump initial condition over the spatial domain $x \in [0, 1]$ at every 0.25 time units in the time interval $t \in [0, 1]$ showing numerical oscillations

During the calculations performed for this section, the addition of numerical oscillations became obvious for calculations with $\nu < 1 \times 10^{-3}$. In Fig. 6.4 such a case is depicted for $\nu = 1 \times 10^{-4}$ (second-order IMEX, $\Delta t = 3 \times 10^{-4}$, $M = 170$). This numerical inaccuracy did not effect the results of the serial study, but it will effect the Parareal study. Because of this only the viscosity $\nu = 0.01$ is considered for the next investigation with the Gaussian bump benchmark.

The reason for the oscillations is the incapability of the chosen spatial discretization of resolving a shock front using only a finite amount of Fourier modes [37]. Such a shock front builds with the benchmark, if the highest initial velocity passes all lower velocities without being reduced by diffusion.

6.3 Comparison Time Stepping Schemes as Coarse Solver

The choice of the coarse solver for Parareal is crucial to the efficiency of the algorithm, as it has to be as cheap and exact as possible the same time. In this section, the three solvers ERK, IMEX, and SLIRK are compared regarding their respective performance with Parareal for the Gaussian bump benchmark. The parameters provided in the benchmark settings in Section 6.1.1 for this comparison were chosen according to the results of the previous serial study.



(a) $\mathcal{C}$: RK

(b) $\mathcal{C}$: IMEX

(c) $\mathcal{C}$: SLIRK

Figure 6.5: Maximal absolute error between the numerical and analytical solution plotted over the time domain for combinations of the second-order IMEX fine solver $\mathcal{F}$ and three different first-order coarse solvers $\mathcal{C}$ of the Parareal algorithm. Given are the errors for the first four Parareal iterations k and the error of $\mathcal{F}$

6.3.1 Coarse Solver of First Order

The comparison is started by investigating the schemes of first-order. In Fig. 6.5 the maximal absolute error of the spatial domain ε_{max} is plotted over the time-slices for the first four Parareal iterations k. Only every fifth time-slice is denoted by a mark to keep the figure clear. In addition to the errors of the iterations, the maximal absolute errors of the fine solver are shown such that a reference is given for a converged solution.

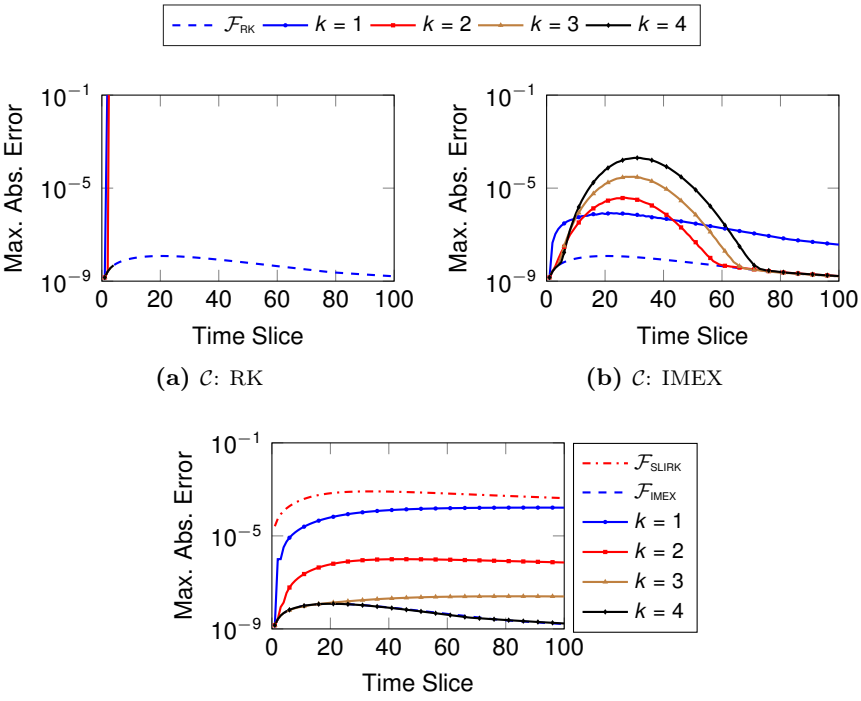
For all three schemes a converged solution for at least the first k time-slices can be observed after k iterations. This basic property of Parareal is well-known, see e.g. [11].

The results of the first-order ERK calculations are provided in Fig. 6.5a. With the chosen parameter setting the coarse ERK solver is not stable according to the serial stability study carried out in the previous section. For all time-slices, which are not the first k, the error is infinite. Since the solution of the fine solver is passed on one time-slice each iteration, convergence is reached eventually after $k = 100$ iterations. For the following investigations the first-order ERK is no longer used, as it provides no useful information.

In contrast to ERK, IMEX is expected to be stable with the chosen coarse solver parameters. This is confirmed for the first-order by the results in Fig. 6.5b. Moreover, the figure shows successive iterations leading to a reduction in the error for all time-slices. The Parareal algorithm finally converges after $k = 8$ iterations, which, considering the total number of time-slices $N_T = 100$, is a good convergence rate.

Finally, the first-order SLIRK has to be discussed. In Fig. 6.5c the error of a fine run with the first-order SLIRK with the time step size $\Delta t = 2 \times 10^{-5}$ is plotted in addition to the other errors. This error curve, being nearly five orders of magnitude higher than the one of the fine IMEX, shows the dependence of the error of SLIRK on the spatial resolution. The error after the first iteration of Parareal with SLIRK is comparable to the one with IMEX. Further iterations reduce the error, even though the error of the coarse solver is still very large due to its dependency on the spatial resolution. This is possible because the spatial resolution error is negated by the prediction-correction step of Parareal in (3.24). The error reduction of SLIRK is even faster than the one of IMEX. This is reflected by the fact that a convergence is already reached after $k = 6$ iterations with this coarse solver.

Overall, IMEX and SLIRK of first-order seem to be very similar in their performance regarding the number of iterations to convergence for this test case. The small advantage of SLIRK regarding the number of iterations is only an advantage if its computational cost is comparable or cheaper than the one of IMEX. This is obviously dependent on the used implementation, as the



(a) $\mathcal{C}$: RK

(b) $\mathcal{C}$: IMEX

(c) $\mathcal{C}$: SLIRK

Figure 6.6: Maximal absolute error between the numerical and analytical solution plotted over the time domain for combinations of the second-order IMEX fine solver $\mathcal{F}$ and three different second-order coarse solvers $\mathcal{C}$ of the Parareal algorithm. Given are the errors for the first four Parareal iterations k and the error of $\mathcal{F}$

evaluation cost of a spatial differential operator is dependent on the spatial discretization and, therefore, the cost is not further discussed here.

6.3.2 Coarse Solver of Second Order

Running the Gaussian bump benchmark with the second-order time-stepping schemes for the coarse solver leads to the results in Fig. 6.6. Depicted is again the error over the time-slices, as it was the case for the first-order comparison.

The second-order ERK again does not yield a stable initial guess for the Parareal algorithm. The serial stability study already foreshadowed this. The errors of each of the four shown iterations in Fig. 6.6a are very large, or immediately infinite for all time-slices except the first k slices. This results in a convergence after $k = 100$ Parareal iterations. Hence, ERK of second-order is also not considered anymore for further investigations.

The calculation with the second-order IMEX yields different results than the ones with first-order. The error after the first iteration visualized in Fig. 6.6b is already a good approximation to the error of the fine solution. Further iterations, however, lead to an increasing error. This happens especially in the first half of the time domain. The increasing error is a sign of reaching a velocity distribution in the intermediate solutions, which leads to unstable behavior of the coarse solver. The fact that the error converges in the later time steps is a result of the good initial approximation and the smaller maximal velocity at those later time points. In addition, the increased errors of the previous time-slices did not propagate to the later time-slices, yet. The effect of the propagation can be seen e.g. at time-slice 60, where at iteration $k = 2$ the error nearly matches the one of the fine solver, but in consecutive iterations the error increases again. In later iterations the increasing errors lead to infinite errors, such that a convergence is found only after $k = 89$ iterations. From this it can be concluded, that stability of a coarse solver for the serial calculation, cf. the serial stability study, is not a sufficient criterion for stability of the coarse solver over all Parareal iterations. This is an example of the instability induced by Parareal.

Since the implementation of the second-order SLIRK method used for this thesis is stable independent of the CFL number, the effects observed with IMEX are not expected to appear. Figure 6.6c confirms SLIRK staying stable during all Parareal iterations. Consecutive iterations lead to decreasing errors. The errors after the first iteration are multiple magnitudes larger than the ones of the second-order IMEX. By keeping the dependency of the error of SLIRK on the spatial resolution in mind this can easily be explained, as the error of the initial guess is much larger for SLIRK than for IMEX. Convergence of the Parareal algorithm is still reached after $k = 5$ iterations. Using a smaller

discretization width in space might lead to a faster convergence for both the first and second-order SLIRK. This option is not explored further within this thesis, as a positive effect of applying SLIRK is already visible with the current version.

Comparing the results of the second-order schemes with the ones of the first-order schemes shows comparable or even better results of the first-order ones. In the further investigation still both orders are used to expand the comparison. The question to be answered is whether the additional cost of the second-order schemes are reasonable.

6.4 Influence of the Coarse Time Step Size on the Convergence of Parareal

Next to the results shown in the previous section, it is important to investigate how significant the time step size of the coarse solver is for the performance of the different time-stepping schemes. Therefore, in this section, the Gaussian bump benchmark is run again with Parareal using multiple different coarse time step sizes. The time step sizes investigated are $\Delta T \in \{3/(2^n) \times 10^{-3} | n \in \{1, \dots, 4\}\}$.

The largely increased number of iterations to convergence for both the ERK method and the second-order IMEX method were attributed to an unstable coarse solver in some of the Parareal iterations. Smaller time step sizes should stabilize the coarse solvers. Here, the number of iterations to convergence is monitored for the different calculations to see the impact of the time step size. The results are provided in Table 6.3.

The table shows convergence of Parareal with the first-order schemes already in a small number of iterations (IMEX: $k = 8$, SLIRK: $k = 6$) for the largest time step size, as stated before. Decreasing the time step size reduces the number of iterations to convergence even further. This complies with the theory of Gander & Hairer [50]. In this reference, the error bound at iteration k of the Parareal algorithm with a coarse solver of order p being proportional to $\Delta T^{p(k+1)}$ was proven.

Table 6.3: Number of iterations needed for convergence with the coarse solvers IMEX and SLIRK in both first- and second-order for varying time step sizes

		Iterations	
Order	Time Step Size	$\mathcal{C}$: IMEX	$\mathcal{C}$: SLIRK
1	3×10^{-3}	8	6
	1.5×10^{-3}	6	5
	7.5×10^{-4}	5	5
	3.75×10^{-4}	4	4
2	3×10^{-3}	89	5
	1.5×10^{-3}	3	4
	7.5×10^{-4}	2	4
	3.75×10^{-4}	2	4

The number of iterations for the first-order SLIRK is $k = 5$ for both $\Delta T = 1.5 \times 10^{-3}$ and $\Delta T = 7.5 \times 10^{-4}$. This is still in line with the mentioned theory, as the convergence is tested against a tolerance and it is possible that in case of $\Delta T = 1.5 \times 10^{-3}$ this tolerance was barely reached, where it was already nearly reached after four iterations with $\Delta T = 7.5 \times 10^{-4}$. For the two smallest time step sizes both IMEX and SLIRK need $k = 4$ iterations to convergence. In these cases it would again be necessary to compare the computational cost of both schemes to decide which is more beneficial. For the larger time steps SLIRK excels IMEX due to its larger stability region, which would allow even larger time step sizes than those tested. In case of $\Delta T = 3 \times 10^{-3}$, where both schemes are stable over all iterations, SLIRK needs two iterations less than IMEX.

With the second-order schemes two interesting things can be seen. First, the second-order IMEX reaches a region of instability after a few iterations for the largest time step size, as shown in the previous section, leading to a convergence only after $k = 89$ iterations. This number decreases drastically to $k = 3$ by halving the time step size used to $\Delta T = 1.5 \times 10^{-3}$ for the coarse solver. In this case the solver is stable for all iterations and yields al-

ready a good approximation of the solution with the initial guess of Parareal. Reducing the time step size further by a factor of two leads to immediate convergence of Parareal after the second iteration (Two are necessary to compute convergence).

The second thing which can be seen is the number of iterations until convergence for SLIRK not dropping below $k = 4$, although the time step size is decreased. The reason for this is the initial approximation quality of the coarse solver being dependent on both the spatial resolution and the temporal discretization. For the smaller time step sizes the error of the initial guess is always bound by the spatial resolution leading to a minimal number of iterations to convergence. Hence, using IMEX would be more efficient in these cases. However, since the idea is to use as large as possible time step sizes for the coarse solver, the largest time step size shows the potential of SLIRK due to its stability. In addition, usually the spatial discretization is not carried out in spectral space making a finer spatial resolution necessary to reach small errors.

6.5 Influence of the SLIRK on the Advective Problem

After the initial comparison of ERK, IMEX, and SLIRK, now it is of interest, how IMEX and SLIRK perform as coarse solvers with smaller viscosities. As described in the sinusoidal waves and saw-tooth benchmark, multiple viscosities are used for the following computations. With these it is possible to probe the advection dominated region where the equation is dominantly hyperbolic. By using the two mentioned benchmarks consisting of manufactured solutions it is possible to circumvent the problems arising from shock build ups.

6.5.1 Sinusoidal Waves Benchmark

First, the sinusoidal waves benchmark is discussed in this subsection. With this benchmark the effect of the coarse solver on a problem with only specific Fourier modes is investigated. The results of each solver are shown separate

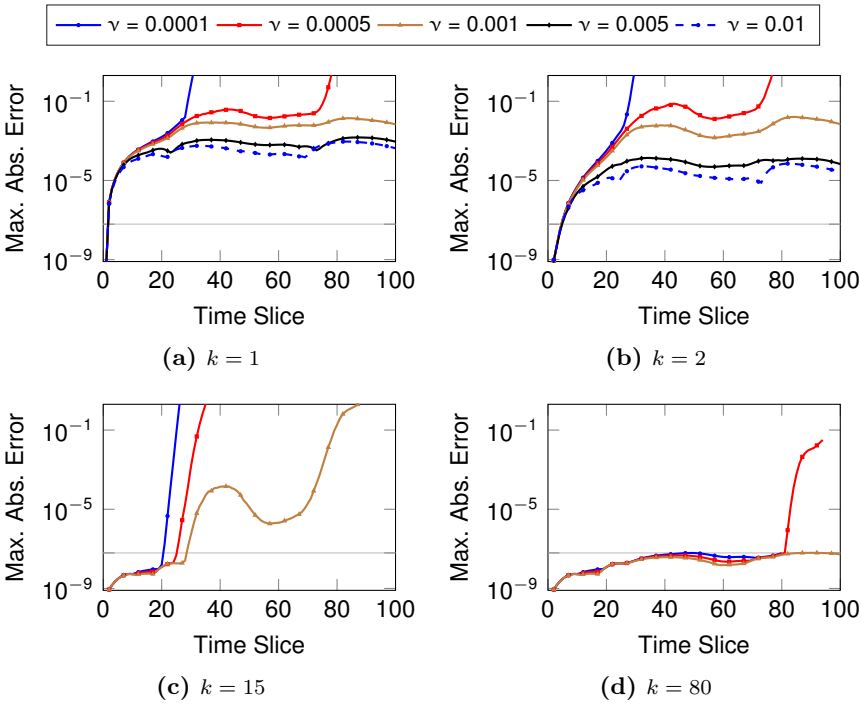


Figure 6.7: Maximal absolute error between numerical and analytical solution of the sinusoidal waves benchmark plotted over the time domain for the Parareal iterations $k \in \{1, 2, 15, 80\}$. The coarse solver $\mathcal{C}$ of the used Parareal algorithm is the first-order IMEX. Given are the errors for chosen viscosities of the set $\nu \in \{n \times 10^{-4}, n \times 10^{-3} | n \in \{1, 2, \ldots, 10\}\}$. The shown constant is the upper bound of the error of the fine solver

at first, before a comparison is carried out based on the number of iterations to convergence of the Parareal algorithm.

First Order IMEX

The first results discussed are the ones of the first-order IMEX calculations. In Fig. 6.7 the results to the Parareal iterations $k \in \{1, 2, 15, 80\}$ are shown. For each iteration the maximal absolute error of the spatial domain is plotted

over the time-slices for the viscosities $\nu \in \{n \times 10^{-4}, n \times 10^{-3} | n \in \{1, 5, 10\}\}$. Only these selected viscosities out of the set defined in the benchmark are shown to make the plot more clear. Furthermore, only every fifth time-slice is denoted with a marker on the plotted curves to reduce the clutter even more. For the same reason, the errors of the fine solver are not shown individually. Plotted is only the bound of these errors, which is $\varepsilon_{\mathcal{F}} < 6.2 \times 10^{-8}$.

After the first iteration, see Fig. 6.7a, the calculations with the two smallest viscosities $\nu \in \{1 \times 10^{-4}, 5 \times 10^{-4}\}$ have large errors leading to infinity. In these cases the coarse solver did diverge. These viscosities will be referred to as advective region. The coarse solvers in the other runs were stable and result in errors which are bound by $\varepsilon < 1.5 \times 10^{-2}$. However, the errors of the two more diffusive calculations with $\nu \in \{5 \times 10^{-3}, 1 \times 10^{-2}\}$ are even one order of magnitude smaller. Further on these viscosities will be called diffusive region.

In the following iteration, shown in Fig. 6.7b, the results of the different viscosities can again be split in the three groups of the advective, diffusive, and in between region. The errors of all calculations decrease for the first 20 time-slices. The results belonging to the advective region still result in infinite errors for the later time-slices and the errors of the time-slices, which are finite, increase. The errors of the calculations of the diffusive region reduce by approximately one order of magnitude for all time-slices. Between the time-slices 20 and 40 and the time-slices 80 and 100 the error of the calculation of the in between region increases slightly. For the time-slices between 40 and 80 the errors stay nearly the same.

The slightly increased error after two iterations for the calculation with $\nu = 1 \times 10^{-3}$ is already the first indication of an unstable coarse solver leading to infinite errors in later iterations. This assumption is confirmed by looking at the results after $k = 15$ iterations in Fig. 6.7c. Here, the calculations of the diffusive region have already converged. All three other calculations have infinite errors. At least for the first 15 time-slices the solution has already converged to the same error as the fine solver, as expected with the Parareal algorithm. The error of the fine runs are not shown for the individual time-slices, but can be estimated due to the mentioned property of Parareal.

The solution of the fine solver is passed through one time-slice per iteration for the two calculations of the advective region, as Fig. 6.7d corroborates. The results after the $k = 80$ iterations show a nearly converged solution for $\nu = 1 \times 10^{-3}$ and convergence for the first 80 time-slices for the other two calculations. Especially, the run with $\nu = 1 \times 10^{-4}$ underlines the convergence of Parareal to the error of the fine solver even with an inefficient coarse solver. The error of the 81st time-slice is already infinite, although the result of time-slice 80 is accurate to the precision of the fine solver.

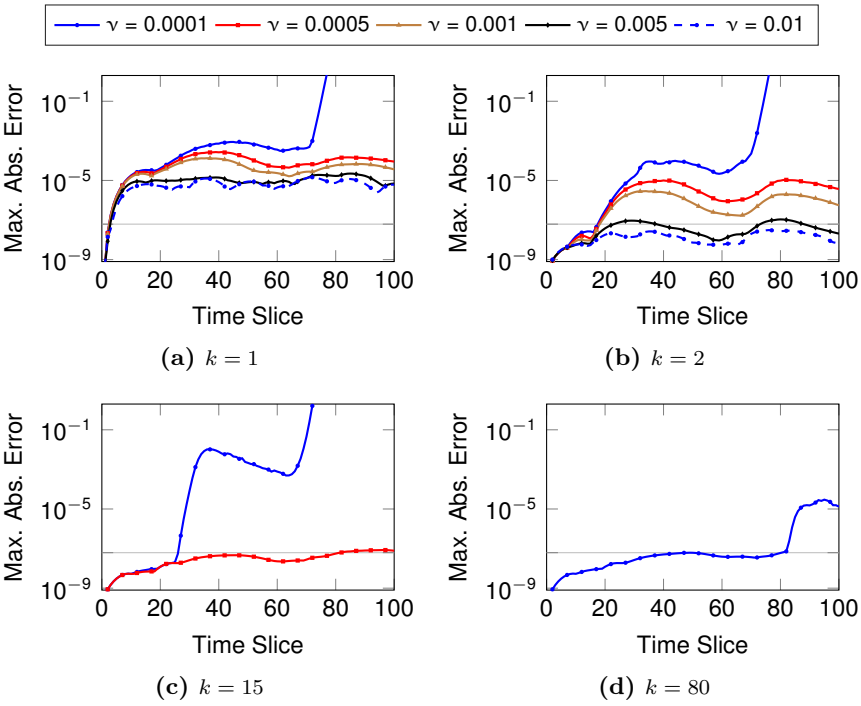
The calculation with $\nu = 1 \times 10^{-3}$ is another example showing a coarse solver which is stable in a serial run not implying a stable behavior throughout all Parareal iterations.

Second Order IMEX

Running the benchmark with the second-order IMEX yields different results. These results are plotted the same way as for the first-order in Fig. 6.8.

After the first iteration only the calculation with $\nu = 1 \times 10^{-4}$ has infinite errors resulting from an unstable coarse solver, see Fig. 6.8a. The second-order schemes lead to a better initial guess for Parareal than the first-order schemes which is reflected by the smaller upper error bound of $\varepsilon < 3 \times 10^{-4}$ for all other calculations. The calculations of the diffusive region are bound by a significantly smaller error with $\varepsilon < 2.3 \times 10^{-5}$.

The next iteration depicted in Fig. 6.8b shows massively reduced errors for the first 20 time-slices. Here, all calculations are very close to convergence. This indicates Parareal having the possibility of being very efficient for short time spans, matching to the theoretical results of [53], where superlinear convergence is shown for short time spans. This sparks the idea for a restarted Parareal approach. The errors of the diffusive region improved by more than two orders of magnitude for all time-slices. For the viscosities $\nu \in \{5 \times 10^{-4}, 1 \times 10^{-3}\}$ the error was reduced by approximately one order of magnitude. The coarse solver of the calculation with $\nu = 1 \times 10^{-4}$ is still divergent.



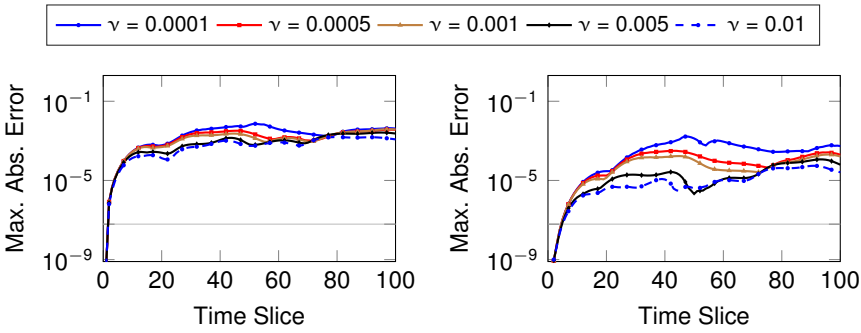
(a) $k = 1$ **(b)** $k = 2$

(c) $k = 15$ **(d)** $k = 80$

Figure 6.8: Maximal absolute error between numerical and analytical solution of the sinusoidal waves benchmark plotted over the time domain for the Parareal iterations $k \in \{1, 2, 15, 80\}$. The coarse solver $\mathcal{C}$ of the used Parareal algorithm is the second-order IMEX. Given are the errors for chosen viscosities of the set $\nu \in \{n \times 10^{-4}, n \times 10^{-3} | n \in \{1, 2, \ldots, 10\}\}$. The shown constant is the upper bound of the error of the fine solver

Only the two calculations of the advective region did not converge after $k = 15$ iterations. Fig. 6.8c shows the calculation with $\nu = 5 \times 10^{-4}$ being already very close to convergence. However, the one with $\nu = 1 \times 10^{-4}$ still has a divergent coarse solver and, hence, converges very slowly.

The maximal velocity in the time-slices after the 80th slice are small enough for the coarse solver of the calculation with $\nu = 1 \times 10^{-4}$ no longer diverging, as depicted in Fig. 6.7d. Still, a convergence of the Parareal algorithm is not achieved.



(a) $k = 1$ **(b)** $k = 2$

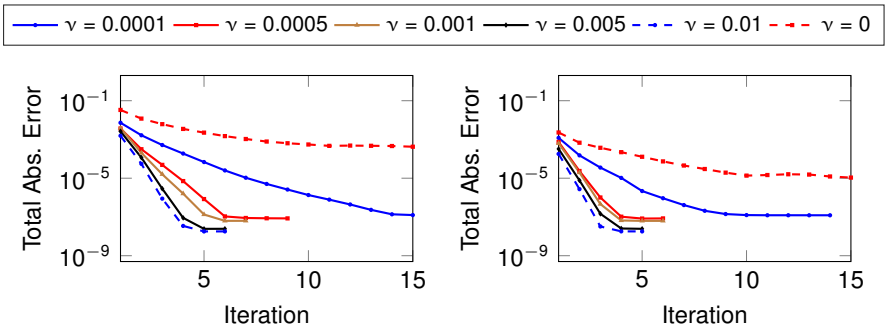
Figure 6.9: Maximal absolute error between numerical and analytical solution of the sinusoidal waves benchmark plotted over the time domain for the Parareal iterations $k \in \{1, 2\}$. The coarse solver $\mathcal{C}$ of the used Parareal algorithm is the second-order IMEX and the first-order SLIRK, respectively. Given are the errors for chosen viscosities of the set $\nu \in \{n \times 10^{-4}, n \times 10^{-3} | n \in \{1, 2, \ldots, 10\}\}$. The shown constant is the upper bound of the error of the fine solver

Overall, comparing the results of first and second-order reveals a higher-order solver leading to a faster convergence of the Parareal algorithm, as it is expected, for all cases where the coarse solver is stable for all iterations. An unstable coarse solver leads to a very slow convergence as it is the case with $\nu = 1 \times 10^{-4}$, where more than $k = 80$ iterations are necessary for convergence.

First Order SLIRK

The results achieved by running the sinusoidal waves benchmark with the first-order SLIRK are shown in Fig. 6.9. Here, only the results of the first two iterations are shown, as further iterations do not provide new input. Otherwise, the plots offer the same graphs as with IMEX.

First, the similarity between the errors after the first iteration for all five calculations can be noted in Fig. 6.11a. None of the calculations shows any sign of an unstable coarse solver. Comparing the calculations with each other reveals slightly increasing errors with a decreasing viscosity. With all errors



(a) First Order **(b)** Second Order

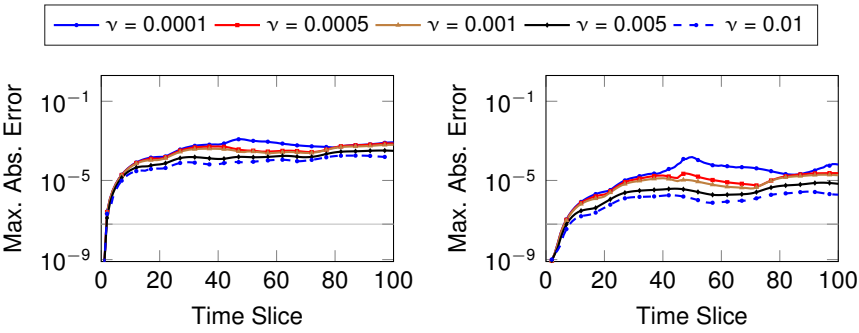
Figure 6.10: Total absolute error in space and time between the numerical and analytical solution of the sinusoidal waves benchmark plotted over the Parareal iterations. The coarse solver $\mathcal{C}$ of the Parareal algorithm is the SLIRK. Given are the errors for chosen viscosities of the set $\nu \in \{0, n \times 10^{-4}, n \times 10^{-3} | n \in \{1, 2, \ldots, 10\}\}$

being smaller than $\varepsilon < 7.5 \times 10^{-3}$, these are similar to the ones obtained with the first-order IMEX.

After the second iteration, the errors of all calculations are reduced compared to the first. Fig. 6.9b shows the amount of the reduction being different for the various viscosities. The errors of calculations with larger viscosities are reduced more, than those of calculations with smaller viscosities. The reduction lies between a factor of approximately five and 100.

This slower reduction of the errors for the smallest viscosity continues until convergence as shown in Fig. 6.10a, where the maximal absolute error over the whole space-time domain is plotted over the Parareal iterations for the selected viscosities. From this graphic the number of iterations until convergence increasing with a decreasing viscosity even though the coarse solver being stable in all cases can be confirmed.

Here, the error decline for a calculation with $\nu = 0$ is shown as well. The result of this computation shows the initial guess getting worse with a decreasing viscosity. The very slow reduction of the error over the iterations



(a) $k = 1$ **(b)** $k = 2$

Figure 6.11: Maximal absolute error between numerical and analytical solution of the sinusoidal waves benchmark plotted over the time domain for the Parareal iterations $k \in \{1, 2\}$. The coarse solver $\mathcal{C}$ of the used Parareal algorithm is the second-order IMEX and the second-order SLIRK, respectively. Given are the errors for chosen viscosities of the set $\nu \in \{n \times 10^{-4}, n \times 10^{-3} | n \in \{1, 2, \ldots, 10\}\}$. The shown constant is the upper bound of the error of the fine solver

leads to a final convergence after $k = 42$ iterations. Therefore, convergence is reached with slightly less iterations than half of the maximum.

Second Order SLIRK

The results to the first two iterations of the calculations done with the second-order SLIRK can be found in Fig. 6.11. Qualitatively the results look similar to the ones obtained with the SLIRK of first-order. However, the errors after the first iteration are smaller being bound by $\varepsilon < 2 \times 10^{-3}$. The difference is not as large as it was between the first and second-order IMEX. The reason lies in the errors dependence on the spatial resolution for SLIRK leading to similar initial guesses. In addition, the spread between the errors of the different calculations after two iterations being not as big as it was with the first-order SLIRK can be noticed. Nevertheless, the calculations with smaller viscosities have always a larger error than those with larger viscosities.

Fig. 6.10b reveals the second-order version of SLIRK leading to faster convergence than the first-order version for all viscosities. Considering the fact

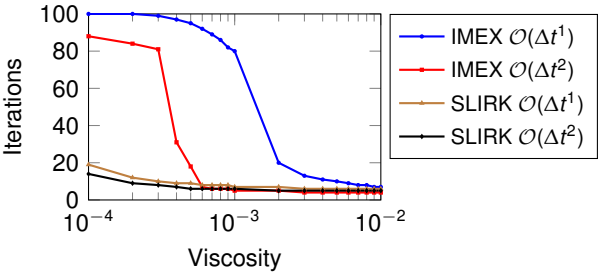


Figure 6.12: Number of Parareal iterations of the sinusoidal waves benchmark plotted over the investigated viscosity set $\nu \in \{n \times 10^{-4}, n \times 10^{-3} | n \in \{1, 2, \ldots, 10\}\}$ for the two coarse solvers for first- and second-order

that $N_T = 100$ time-slices are used, a convergence after 14 iterations, as it is the case for $\nu = 1 \times 10^{-4}$, is still a significant improvement over a serial calculation. Depending on the desired final accuracy, the tolerance used to check for convergence for the Parareal algorithm might be relaxed such that convergence is reached even faster.

Again, the results of an additional computation of the inviscid Burgers equation ($\nu = 0$) are shown in Fig. 6.10b. Compared to the first-order the initial errors of the second-order calculations deviate not as much for decreasing viscosities. The inviscid case is in line with the previous observation of faster convergence for second-order methods over first-order methods. A final convergence is reached after $k = 36$ iterations, which leaves room for some speedup.

Number of Iterations to Convergence

The previous results already gave an indication of the reproducibility of the problem with a bad efficiency of Parareal for small viscosities. Calculations with smaller viscosities needed a larger number of iterations for convergence. In Fig. 6.12 the number of iterations until convergence of the Parareal algorithm is plotted over the whole viscosity set of the benchmark for both the IMEX and the SLIRK scheme of first- and second-order.

For all four time-stepping scheme combinations the number of iterations until convergence is low in cases of diffusion dominated problems. For viscosities $\nu \geq 6 \times 10^{-4}$ IMEX of second-order needs about the same number of iterations until convergence as the second-order SLIRK. In this region, the IMEX has to be seen superior to the SLIRK method, as the additional overhead necessary for the second-order SLIRK regarding the communication does not yield any benefit.

With smaller viscosities the number or iterations necessary for convergence for the IMEX scheme of both orders increases drastically, as at some point the coarse solver becomes unstable during the iterations. Here, the benefit of the SLIRK method comes into play. Due to its stability regardless of the CFL number the number of iterations to convergence increases only moderately for the smaller viscosities. For $\nu = 1 \times 10^{-4}$ the first-order SLIRK converges after $k = 19$ iterations and the second-order already after $k = 14$ iterations. It has to be investigated, whether the lower number of iterations necessary with the second-order SLIRK is actually better than the cheaper to evaluate first-order SLIRK.

6.5.2 Saw-Tooth Function

The sinusoidal waves benchmark gave already a first impression of the potential of the SLIRK method. With the saw-tooth benchmark, now a setting is tested, where all available spectral modes contribute to the solution. Due to this, in each time step from t_n to t_{n+1} all modes of the unknown at time t_n are non-zero. In the previous benchmark only two selected modes were non-zero. The present benchmark is in that regard closer to a calculation without a manufactured solution. Again, the time step size of the coarse solver was chosen such that a serial stability study shows comparable results to the one provided in Section 6.2.

First Order IMEX

The results achieved by using the first-order IMEX as the coarse solver are shown in Fig. 6.13. Here, the maximal absolute error of the spatial domain

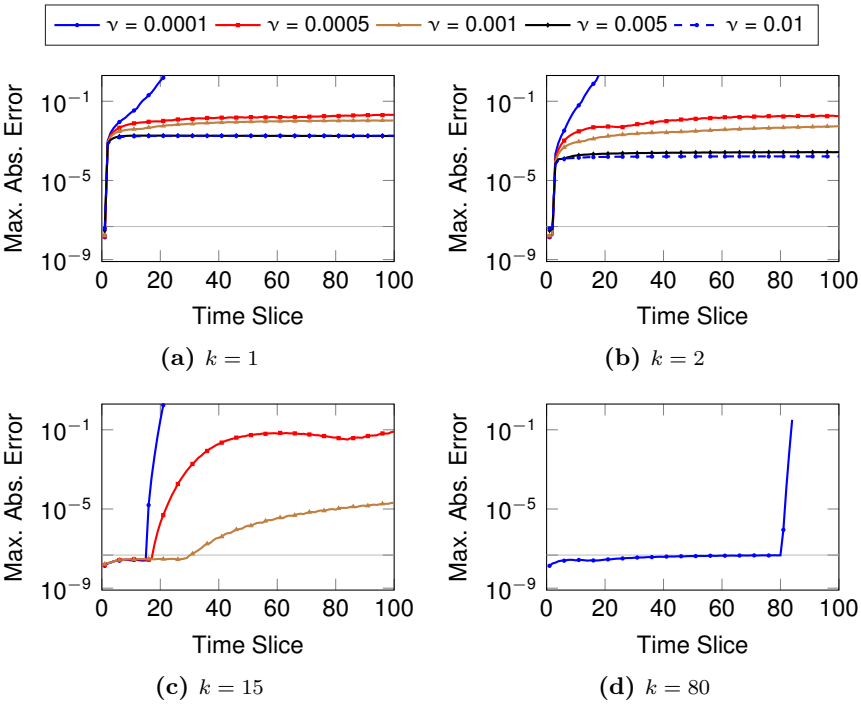


Figure 6.13: Maximal absolute error between numerical and analytical solution of the saw-tooth benchmark plotted over the time domain for the Parareal iterations $k \in \{1, 2, 15, 80\}$. The coarse solver $\mathcal{C}$ of the used Parareal algorithm is the second- and first-order IMEX, respectively. Given are the errors for chosen viscosities of the set $\nu \in \{n \times 10^{-4}, n \times 10^{-3} | n \in \{1, 2, \ldots, 10\}\}$. The shown constant is the upper bound of the error of the fine solver

is plotted over the time-slices for the iterations $k \in \{1, 2, 15, 80\}$ and selected viscosities $\nu \in \{n \times 10^{-4}, n \times 10^{-3} | n \in \{1, 5, 10\}\}$ of the viscosity set defined for the benchmark. The same set up as before is used to keep the plots as clear as possible. The errors of the fine solver are bound by $\varepsilon_{\mathcal{F}} < 4.7 \times 10^{-8}$. This bound is indicated in the plot as well.

For the first iteration the errors are depicted in Fig. 6.13a. Only the smallest viscosity $\nu = 1 \times 10^{-4}$ leads to an unstable coarse solver. All other calculations have a stable coarse solver and their errors are bound by at least $\varepsilon < 2.1 \times$

10^{-2}. The errors of the calculations with $\nu \geq 5 \times 10^{-3}$ are even one order of magnitude smaller and bound by $\varepsilon < 1.9 \times 10^{-3}$. Compared to the previous benchmark, the errors are nearly constant for all time-slices. The reason for this is the constant maximal velocity for all time-slices.

Fig. 6.13b shows the increasing difference between the error bounds of the various viscosities. The already smaller errors achieved with $\nu \geq 5 \times 10^{-3}$ were decreased by approximately one order of magnitude from iteration one to two. The iteration update for the calculation with $\nu = 1 \times 10^{-3}$ decreased the error bound only by a factor between two and three. The run with $\nu = 5 \times 10^{-3}$ updated in the second iterations to a slightly smaller error bound, although the errors increased for some time-slices. This underlines the problem Parareal has with handling advection dominated problems.

After $k = 15$ iterations, the calculations of the diffusive regime $\nu \in \{5 \times 10^{-3}, 1 \times 10^{-2}\}$ did already converge, see Fig. 6.13c. The errors of the calculation with $\nu = 1 \times 10^{-3}$ slowly, but steadily decrease, still having an upper bound of $\varepsilon < 2.1 \times 10^{-5}$. A further increase in the errors belonging to the calculation with $\nu = 5 \times 10^{-4}$ can be observed for the later time-slices compared to the first and second iteration showing an unstable behavior of the coarse solver. The calculation with $\nu = 1 \times 10^{-4}$ converged up to the 15th time-slice. Over the following time-slices the errors increase drastically up to infinity due to the unstable coarse solver.

The only calculation not converged after $k = 80$ iterations is the one of the smallest viscosity $\nu = 1 \times 10^{-4}$. Since its coarse solver is not stable from the start, a convergence is not expected earlier than the 100th iteration when the fine solution is passed through serially.

Second Order IMEX

Applying the second-order IMEX as the coarse solver to the smoothened sawtooth benchmark produces different results to the ones seen with either the first-order IMEX, or the sinusoidal waves benchmark. The results are provided in Fig. 6.14, where once more the maximal absolute error of the spatial domain

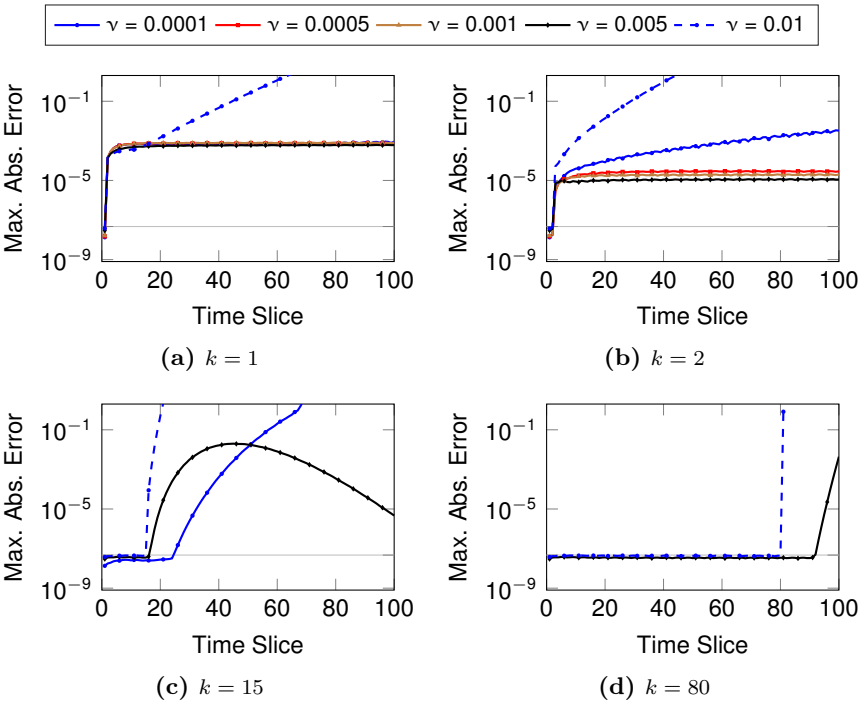


Figure 6.14: Maximal absolute error between numerical and analytical solution of the saw-tooth benchmark plotted over the time domain for the Parareal iterations $k \in \{1, 2, 15, 80\}$. The coarse solver $\mathcal{C}$ of the used Parareal algorithm is the second-order IMEX. Given are the errors for chosen viscosities of the set $\nu \in \{n \times 10^{-4}, n \times 10^{-3} | n \in \{1, 2, \ldots, 10\}\}$. The shown constant is the upper bound of the error of the fine solver

is plotted over the time-slices for the iterations $k \in \{1, 2, 15, 80\}$ and selected viscosities $\nu \in \{n \times 10^{-4}, n \times 10^{-3} | n \in \{1, 5, 10\}\}$.

After one Parareal iteration, in Fig. 6.14a only one calculation can be identified with a diverging coarse solver. In contrast to the previous results, the calculation with infinite errors is the one with the largest viscosity $\nu = 1 \times 10^{-2}$. As mentioned before, the second-order IMEX is not unconditionally stable in the diffusive region, as an explicit evaluation of the diffusive term is carried out in the scheme. This is also the reason for the unstable coarse solver, here,

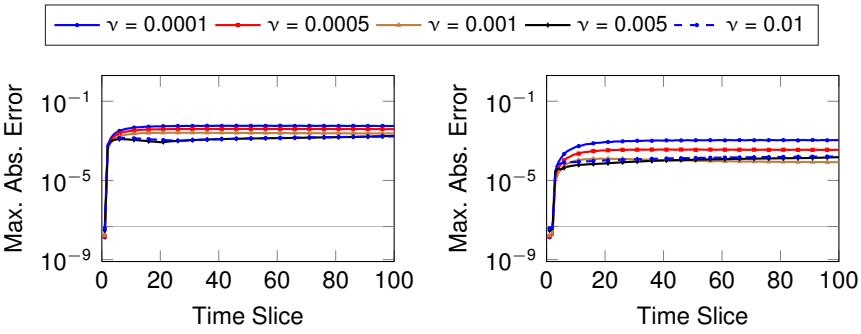
especially since this benchmark has large curvatures leading to large diffusive velocities. The errors of all other calculations are very similar and bound by $\varepsilon < 7.9 \times 10^{-4}$.

The results of the next iteration provided in Fig. 6.14b show a comparable reduction of the error for the viscosities $\nu \in \{5\times10^{-4}, 1\times10^{-3}, 5\times10^{-3}\}$. For these the errors are now bound by $\varepsilon < 3.0 \times 10^{-5}$ which is an improvement of more than one order of magnitude. The coarse solver of the calculation with $\nu = 1 \times 10^{-4}$, however, produces increasing errors for the later time-slices due to instabilities. The errors of the calculation with $\nu = 1 \times 10^{-2}$ increase as well from iteration one to two, such that the infinite error is found already at earlier time-slices.

The plot belonging to iteration $k = 15$ in Fig. 6.14c confirms the coarse solver of the calculation with $\nu = 1 \times 10^{-4}$ operating in its unstable region now showing infinite errors for the later time-slices. In contrast to the other unstable coarse solvers, the errors of this calculation do not increase as fast such that the infinite errors are only many time-slices after the one with the same number as the current iteration. The calculations with $\nu \in \{5\times10^{-4}, 1\times 10^{-3}\}$ have already converged. Although the run with $\nu = 5 \times 10^{-3}$ seemed to be stable and fast converging, the error increasing again can be seen here resulting in an error bound of $\varepsilon < 2.0 \times 10^{-2}$.

The calculation belonging to $\nu = 1\times10^{-4}$ is already converged after $k = 80$ iterations, as depicted in Fig. 6.14d. This is due to the fact that in earlier iterations the increased errors due to the unstable coarse solver did only appear in later time-slices. With this the correct solution was already found for more time-slices than the run number of iterations being able to correct the infinite errors already at a smaller iteration number. The results of the computations with the larger viscosities $\nu \in \{5 \times 10^{-3}, 1 \times 10^{-2}\}$ show still large errors implying an unstable coarse solver. This instability is based on the partially explicit handling of the diffusive term.

Both the first and the second-order IMEX show again a coarse solver which is stable for the initial guess not necessary staying stable for all iterations. Therefore, solvers with larger stability regions are expected to be more efficient with Parareal regarding the number of iterations until convergence.



(a) $k = 1$ **(b)** $k = 2$

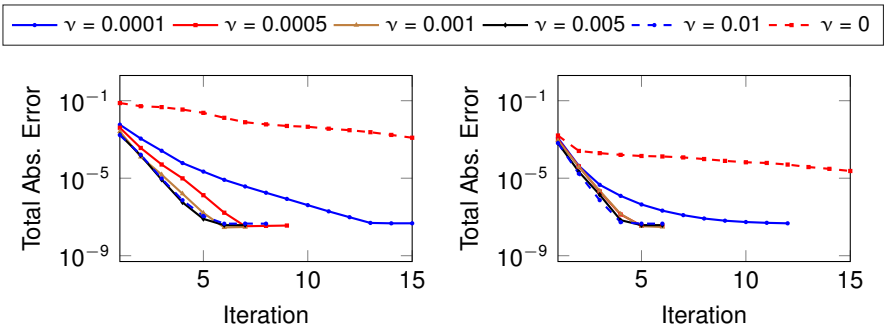
Figure 6.15: Maximal absolute error between numerical and analytical solution of the saw-tooth benchmark plotted over the time domain for the Parareal iterations $k \in \{1, 2, 5\}$. The coarse solver $\mathcal{C}$ of the used Parareal algorithm is the second-order IMEX and the first-order SLIRK, respectively. Given are the errors for chosen viscosities of the set $\nu \in \{n \times 10^{-4}, n \times 10^{-3} | n \in \{1, 2, \ldots, 10\}\}$. The shown constant is the upper bound of the error of the fine solver

First Order SLIRK

In Fig. 6.15 the maximal absolute errors over the spatial domain are plotted over the time-slices for the calculations run with the first-order SLIRK as the coarse solver. As it was done with the previous test case, the errors are shown only for the first two iterations.

None of the coarse solvers shows a sign of instability after the first iteration as depicted in Fig. 6.15a. The errors of the different calculations do not differ much. The calculation with $\nu = 1 \times 10^{-4}$ has the largest errors bound by $\varepsilon < 5.8 \times 10^{-3}$. The upper bound of the errors for the calculation with $\nu = 1 \times 10^{-2}$ is with $\varepsilon < 1.7 \times 10^{-3}$ the smallest. With this, these errors after the initial iteration are comparable to the ones achieved with the first-order IMEX method after one iteration.

The error reduction from the first to the second iteration is not the same for all computations. Fig. 6.15b shows the errors of the calculations with $\nu \geq 1 \times 10^{-4}$ converging with approximately the same rate. However, the errors of the other two runs improve less. Nevertheless, the errors of these



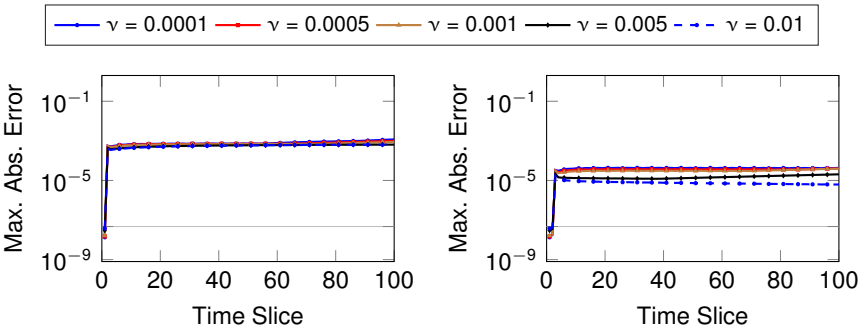
(a) First Order **(b)** Second Order

Figure 6.16: Total absolute error in space and time between the numerical and analytical solution of the saw-tooth benchmark plotted over the Parareal iterations. The coarse solver $\mathcal{C}$ of the Parareal algorithm is the SLIRK. Given are the errors for chosen viscosities of the set $\nu \in \{0, n \times 10^{-4}, n \times 10^{-3} | n \in \{1, 2, \ldots, 10\}\}$

two calculations still recede. Overall, the errors at this point are smaller than they were with the first-order IMEX.

The tendencies observed in the first two iterations show as well in Fig. 6.16a, where the maximal absolute error of the space-time domain is plotted over the number of iterations. The cases with $\nu \geq 1 \times 10^{-3}$ have a comparable error reduction over the iterations and converge in a similar number of iterations. Not far off is the error reduction of the calculation with $\nu = 5 \times 10^{-4}$ which converges with only one or two more iterations. The calculation with $\nu = 1 \times 10^{-4}$, however, needs significantly more iterations to converge. Convergence is still reached in $k = 16$ iterations, which is not too much considering the total number of time-slices is $N_T = 100$ and the tolerance for convergence being set to $tol = 1 \times 10^{-8}$.

For this benchmark the inviscid case was run as well to investigate its error reduction. The initial error of the inviscid case is an order of magnitude larger than the viscid cases shown. In addition the error reduction over the iterations is small, leading to a final convergence after $k = 56$ iterations. Compared to the previous benchmark it can be seen that the more realistic setup, where all



(a) $k = 1$ **(b)** $k = 2$

Figure 6.17: Maximal absolute error between numerical and analytical solution of the saw-tooth benchmark plotted over the time domain for the Parareal iterations $k \in \{1, 2, 5\}$. The coarse solver $\mathcal{C}$ of the used Parareal algorithm is the second-order IMEX and the second-order SLIRK, respectively. Given are the errors for chosen viscosities of the set $\nu \in \{n \times 10^{-4}, n \times 10^{-3} | n \in \{1, 2, \ldots, 10\}\}$. The shown constant is the upper bound of the error of the fine solver

modes have a non-zero value, needs more iterations to converge and, hence, yields a not so good speedup potential.

Second Order SLIRK

Comparable to the results obtained with the first-order SLIRK as the coarse solver are those with the second-order SLIRK shown in Fig. 6.17. Still, some differences can be observed.

The errors after the first iteration are bound by $\varepsilon < 1.5 \times 10^{-3}$ for all calculations and the difference between the errors of the calculations is negligible, see Fig. 6.17a. This suggests the error of the initial guess possibly being bound by the spatial resolution.

After the second iteration the errors improve in the range between one and two orders of magnitude as depicted in Fig. 6.17b. Again, the more diffusive setups have the larger error improvement than the advection dominated setups. The upper bounds of the errors are $\varepsilon < 4.4 \times 10^{-5}$ and $\varepsilon < 1.8 \times 10^{-5}$ for the calculation with $\nu = 1 \times 10^{-4}$ and $\nu = 1 \times 10^{-2}$, respectively.

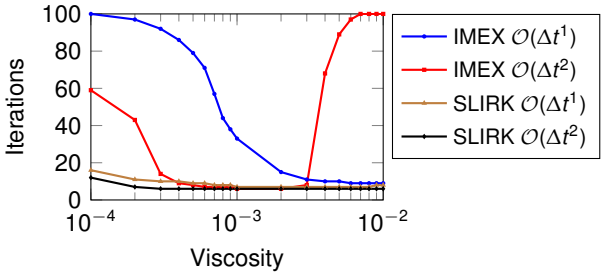


Figure 6.18: Number of Parareal iterations of the saw-tooth benchmark plotted over the investigated viscosity set $\nu \in \{n \times 10^{-4}, n \times 10^{-3} | n \in \{1, 2, \ldots, 10\}\}$ for the two coarse solvers for first- and second-order

The difference in the improvement of the error between the calculations is not as distinct for the first few iterations with the second-order SLIRK, as it was with the first-order SLIRK. In Fig. 6.16b a plain difference in the gradient can not be seen until the third iteration. Afterwards, the convergence rate of the case with $\nu = 1 \times 10^{-4}$ is clearly slower than the rate of all other calculations. However, the convergence is reached after $k = 12$ iterations, which is nearly a factor of ten smaller than the worst case scenario of $k = 100$ iterations showing still a speedup possibility.

The additionally plotted inviscid case shows a good error reduction from iteration one to two, but for further iterations the reduction is very slow. This calculation finally converges after $k = 56$ iterations, as it was also the case with the first-order run. Comparing these two runs, shows the first-order run starting with an error which is nearly two magnitudes larger than the one of the second-order run. Hence, the first-order SLIRK seems to reduce the error faster than the second-order version. Nevertheless, for both cases the number of iterations to convergence leaves only possibilities for small speedups.

Number of Iterations to Convergence

The number of iterations until convergence for all viscosities of the benchmark are depicted in Fig. 6.18. It shows the first-order IMEX behaving as expected. For diffusion dominated problems only a few number of iterations are neces-

sary for convergence, contrasted with advection dominated cases, where the fine solution is passed through serially. The increasing number of iterations until convergence for advection dominated problems can also be observed for the second-order IMEX. With this scheme a special case is found with diffusion dominated problems. These should be beneficial for Parareal, but due do to instabilities of the solver no convergence is found earlier than after $k = 100$ iterations.

Convergence with the SLIRK schemes is not impacted that drastically over the investigated viscosity set. In diffusion dominated cases between $6 \leq k \leq 8$ iterations are necessary for convergence and in advection dominated cases between $12 \leq k \leq 16$ iterations. The second-order method needs always less iterations, than the first-order one. Due to the larger overhead especially in communication of the second-order SLIRK it might be faster to use the first-order SLIRK. Another possibility is to circumvent the additional overhead is to use the second-order IMEX, in cases, where both schemes need a similar number of iterations to converge.

Overall, the benefits of the more stable SLIRK formulation can be seen clearly in these results. Even though, the number of iterations until convergence increases with a decreasing viscosity, only a relatively small number of iterations is necessary to reach convergence.

6.6 Influence of Wave Propagation Characteristics

The previous section has shown the choice of the coarse solver having a significant impact on the convergence behavior of Parareal. This raises the question how these two schemes compare to each other. Ruprecht & Krause have shown in [86] a quick convergence of Parareal as long as a wave is placed near its correct position by the coarse solver, even with strong numerical diffusion. In [85] Ruprecht expanded this research by investigating the wave propagation characteristics of Parareal with the linear advection-diffusion equation. He showed the observed instability within Parareal being induced by an er-

roneous prediction of the phase of the propagated wave by the coarse solver. Following this result, the two schemes are compared here with respect to the phase and amplitude errors of the propagated waves to check whether Ruprecht's results are transferable to the non-linear case.

An analytical notation of the discrete dispersion relation, as it was done in [85], is not practical in the non-linear case. Because of this, the phase and amplitude error evolution by Parareal is carried out in a numerical experiment. For the computations the smoothened saw-tooth benchmark is used with the modifications stated in the following subsections. Before the influence of the Parareal iterations on the errors is investigated, the propagation errors of the coarse solvers during a serial run are looked at.

6.6.1 Wave Propagation Characteristics of IMEX

First, the propagation errors of the IMEX method are investigated. For this the smoothened saw-tooth benchmark is run in serial using the time step of the coarse solver. The viscosity set is reduced to the viscosities $\nu \in \{1 \times 10^{-n} | n \in \{2, 3, 4\}\}$. The phase and amplitude errors of the waves are computed by subtracting the analytical phase and amplitude from their numerical counterparts. Therefore, positive errors in the amplitude are a numerical overestimation and negative errors an underestimation. In case of the phase errors, positive errors are an acceleration and negative errors a deceleration of the wave transport.

The results belonging to the first-order IMEX with $\nu = 1 \times 10^{-3}$ are shown in Fig. 6.19. Here, the phase and the relative amplitude errors of the propagated waves are plotted over the spectral modes of the spatial discretization for three different time points. These time points are after one single time step ($t = 0.005$) and after multiple time steps at $t = 0.25$ and $t = 0.75$. Since the phase is 2π-periodic, the phase error is plotted in the range from $-\pi$ to π. There are only $M = 85$ spectral modes shown, since the solution in physical space is real valued and the remaining modes are redundant.

After one time step the relative amplitude errors for the first half of the spectral modes up to $m = 40$ are very small. Higher modes show higher

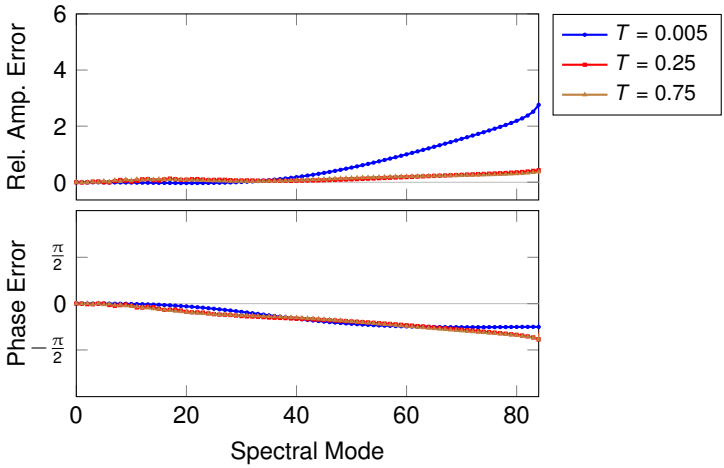


Figure 6.19: Relative amplitude and phase errors of the propagated waves plotted over the spectral modes for one ($t = 0.005$) and multiple time steps ($t = 0.25$ and $t = 0.75$) with the first-order IMEX. The viscosity is set to $\nu = 0.001$

errors. The maximal relative error is $\varepsilon_{rel} = 2.8$. A numerical deceleration of the wave transport is found in all modes. Higher modes show again larger phase errors, but for modes higher than $m > 60$ the errors are comparable to each other with a phase error of $\varepsilon_\Phi = -0.8$ rad.

Both the relative amplitude and the phase errors of the two other time points shown are very similar to each other. The maximal relative amplitude error is much smaller than after the first time step with $\varepsilon_{rel} = 0.43$. The amplitude errors for all modes higher than $m > 32$ are smaller than those after the first time step. For all other modes the relative amplitude error is larger. A nearly linear increase can be seen for the phase error with increasing modes. The maximal phase error of $\varepsilon_\Phi = -1.2$ rad is 1.5 times larger than the one after the first time step.

The results for the second-order IMEX with $\nu = 1 \times 10^{-3}$ are depicted in Fig. 6.20. Here, the errors of all three time points are similar. The relative amplitude errors are small for all spectral modes up to the 60th mode. For the higher modes large relative amplitude errors can be observed, where again a

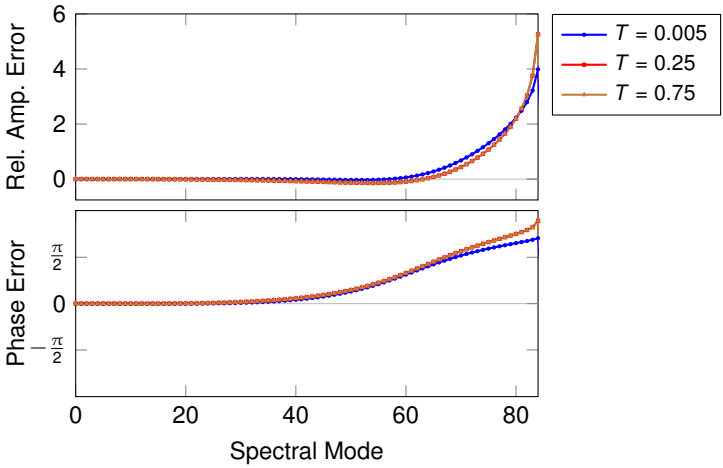


Figure 6.20: Relative amplitude and phase errors of the propagated waves plotted over the spectral modes for one ($t = 0.005$) and multiple time steps ($t = 0.25$ and $t = 0.75$) with the second-order IMEX. The viscosity is set to $\nu = 0.001$

higher mode has a larger error. After the first time step the maximal relative amplitude error is $\varepsilon_{rel} = 4.0$. At the other two time steps the maximal error is even larger reaching $\varepsilon_{rel} = 5.3$. The phase errors are small for the first $M = 40$ spectral modes, but show an acceleration for all modes. For the modes $m > 40$ large errors can be seen with a maximal phase error of $\varepsilon_\Phi = 2.2$ rad after the first time step and $\varepsilon_\Phi = 2.8$ rad for the other two time points for the highest mode.

These results already show both IMEX solvers having significant errors in the phase, especially for the higher modes. The efficiency deficit of these with Parareal shown before supports the assumption of [86] that it is important to correctly place the waves for a good efficiency of Parareal. In addition, the results indicate the irrelevance of whether the position of the wave is over- or underpredicted.

Fig. 6.21 shows the phase and amplitude errors of the propagated waves for the second-order IMEX method with $\nu = 1 \times 10^{-2}$, which is a diverging calculation. The divergence is clearly visible in the amplitude error, which

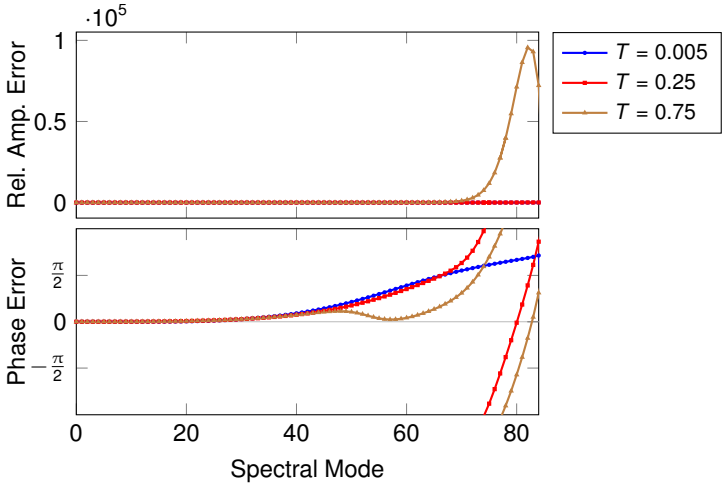


Figure 6.21: Relative amplitude and phase errors of the propagated waves plotted over the spectral modes for one ($t = 0.005$) and multiple time steps ($t = 0.25$ and $t = 0.75$) with the second-order IMEX to show instability. The viscosity is set to $\nu = 0.01$

reaches $\varepsilon_{rel} = 7.2 \times 10^4$ at the time point $t = 0.75$. In the phase error plot, the phase errors after the first time step being comparable to those shown before in Fig. 6.20 can be seen. However, further iterations increase the phase errors in the higher spectral modes so much, that the higher modes show errors all over the 2π-periodicity. This means the position of the higher mode waves being predicted up to one period and more ahead of their correct position. More time steps, in this case from $t = 0.25$ to $t = 0.75$, decrease the phase error in the higher modes, but as described before, increase the amplitude error drastically.

6.6.2 Wave Propagation Characteristics of SLIRK

The same calculations as described in the previous subsection were carried out with the SLIRK method. The results for the first-order version are provided in Fig. 6.22. The plot is build the same as it was with IMEX. Here, the relative

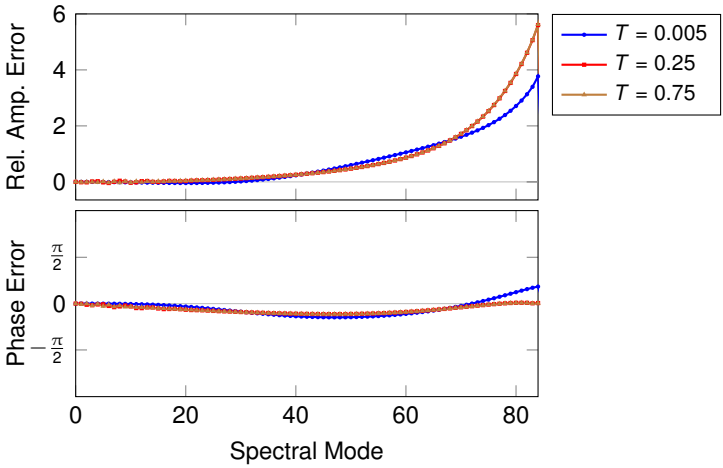


Figure 6.22: Relative amplitude and phase errors of the propagated waves plotted over the spectral modes for one ($t = 0.005$) and multiple time steps ($t = 0.25$ and $t = 0.75$) with the first-order SLIRK. The viscosity is set to $\nu = 0.001$

amplitude error of the zeroth spectral mode is set to zero for all results. Calculating the relative error with respect to the correct solution of zero is not possible. Therefore, the error produced by the missing mass conservation of the semi-Lagrangian formulation is not shown. The errors of the missing mass conservation are discussed instead by their absolute values.

After one time step, higher Fourier modes have larger relative amplitude errors for the SLIRK of first-order. The errors are especially large in the higher modes. The largest error is $\varepsilon_{rel} = 3.8$. The absolute error at the zeroth mode induced by the missing mass conservation is below the machine precision after the first time step. In contrast to the results of IMEX, the phase errors of the first-order SLIRK show both acceleration and deceleration. Deceleration is found for the spectral modes between $m = 3$ and $m = 71$. All other modes have a numerical acceleration. The phase errors are bound by $|\varepsilon_\Phi| < 0.58$ rad, which is similar to the first-order IMEX.

The errors for the other two time points are visually indistinguishable from each other. The multiple time steps lead to relative amplitude errors which are

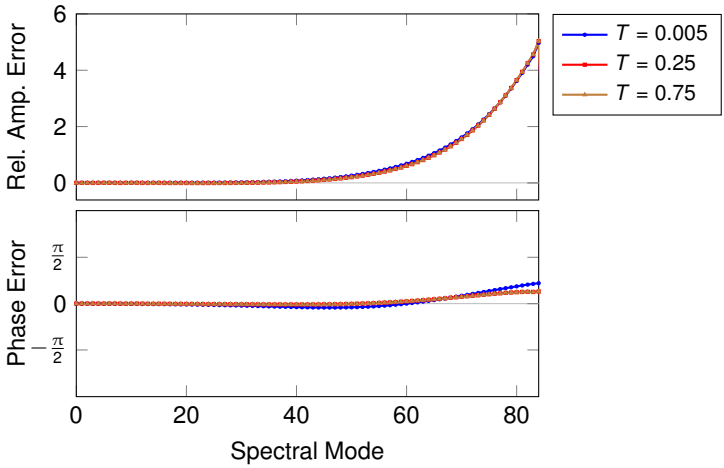


Figure 6.23: Relative amplitude and phase errors of the propagated waves plotted over the spectral modes for one ($t = 0.005$) and multiple time steps ($t = 0.25$ and $t = 0.75$) with the second-order SLIRK. The viscosity is set to $\nu = 0.001$

comparable to the one after one time step for the first $M = 68$ spectral modes. For higher modes the errors did increase to a maximum of $\varepsilon_{rel} = 5.6$. Due to the missing mass conservation the absolute error at the zeroth mode increases to $\varepsilon = 9.0 \times 10^{-5}$ and $\varepsilon = 3.1 \times 10^{-4}$ for $t = 0.25$ and $t = 0.75$, respectively. The phase errors decreased slightly for the first $M = 25$ modes. For the modes between $m = 25$ and $m = 70$ The phase errors increased slightly, but the biggest change is the phase error being reduced by more than one order of magnitude for the modes larger than $m > 70$. Overall the maximal phase error is now $\varepsilon_\Phi = -0.36$ rad and appearing at the mode $m = 46$.

Fig. 6.23 shows the errors of the propagated waves for the second-order SLIRK calculations. The relative amplitude errors of all three investigated time points are nearly the same. Once more, the errors of the higher spectral modes are the largest. This is in particular true for the modes $m > 40$. The maximal relative amplitude error for all three time points is $\varepsilon_{rel} = 5.0$ at the highest mode. Again, the absolute amplitude error at the zeroth mode increases with the number of time steps due to the missing mass conservation

from $\varepsilon = 2.6 \times 10^{-4}$ after one time step to $\varepsilon = 9.3 \times 10^{-4}$ at $t = 0.75$. The phase errors after one time step are comparable to those seen with the first-order SLIRK. The modes in the decelerating part, however, have smaller errors than in the first-order case. The highest phase error is found with the highest mode and is $\varepsilon_\Phi = 0.69$ rad. Multiple time steps as carried out to reach the other two time points reduce the phase errors for all spectral modes. The decrease of the errors in the high spectral modes is not as large, as it was with the first-order method, leaving a maximal error of $\varepsilon_\Phi = 0.41$ rad.

Compared to the phase errors of IMEX, SLIRK shows a good approximation of the wave position in all modes. This further underlines the importance of predicting the phase correctly with Parareal, as SLIRK showed better efficiency than IMEX. Furthermore, the amplitude error can not be as important to the Parareal convergence, because both methods have comparable relative amplitude errors with a slight advantage for IMEX.

The fact that number of iterations until convergence increased for the coarse SLIRK solver with a decreasing viscosity suggests the phase error of the propagated waves for the SLIRK schemes increasing with a decreased viscosity. Therefore, the phase errors for the first- and second-order SLIRK are shown in Fig. 6.24 for the calculations with $\nu = 1 \times 10^{-4}$. After one time step the errors look comparable to the ones with the larger viscosity. An increasing number of time steps leads especially in the case of the first-order SLIRK to very different errors. At time point $t = 0.25$ the phase errors are similar to those found with the larger viscosity, but the line connecting the errors in the plot has an additional oscillation on top of its counterpart. This oscillation gets even bigger for $t = 0.75$. At that time point phase errors are between four and five times larger than those with with the larger viscosity $\nu = 1 \times 10^{-3}$.

Multiple time steps with the second-order SLIRK show increased phase errors for the first $M = 76$ spectral modes for both one time step and multiple time steps compared with those of the calculations with $\nu = 1 \times 10^{-3}$. The errors for higher modes are decreased with respect to one time step. Comparing the high modes of the two calculations with different viscosity, shows only the two highest modes having smaller phase errors with $\nu = 1 \times 10^{-4}$, where as for all other modes the errors are larger.

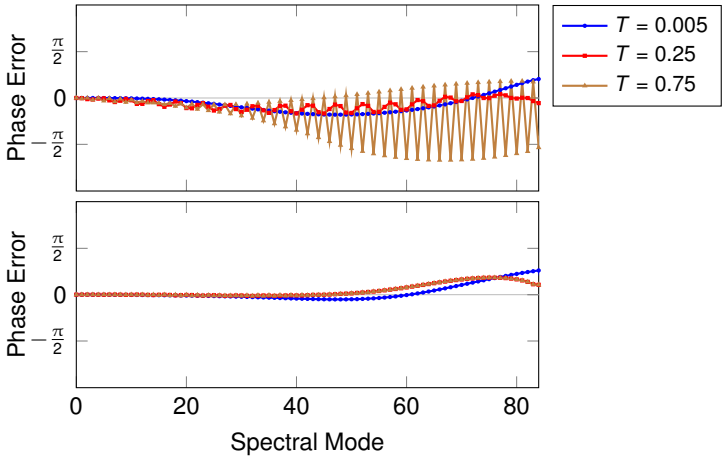


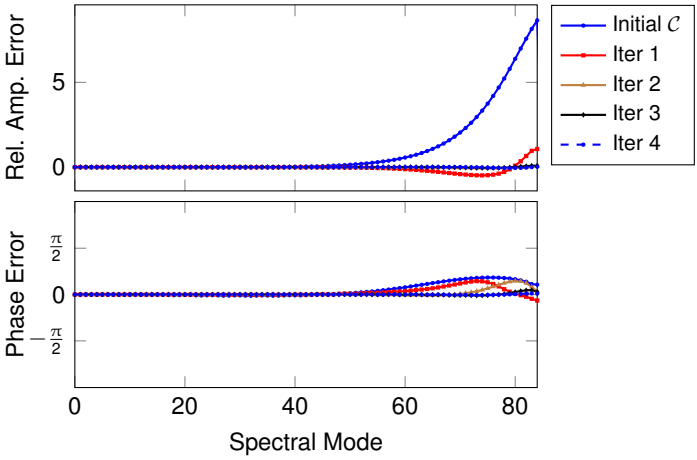
Figure 6.24: Phase errors of the propagated waves plotted over the spectral modes for one ($t = 0.005$) and multiple time steps ($t = 0.25$ and $t = 0.75$) with first- and second-order SLIRK (First-order: top, second-order: bottom). The viscosity is set to $\nu = 0.0001$

Since for both orders of the SLIRK most of the phase errors in the higher modes are larger with the smaller viscosity, the larger number of iterations to convergence found in Sec. 6.5.2 is reasonable.

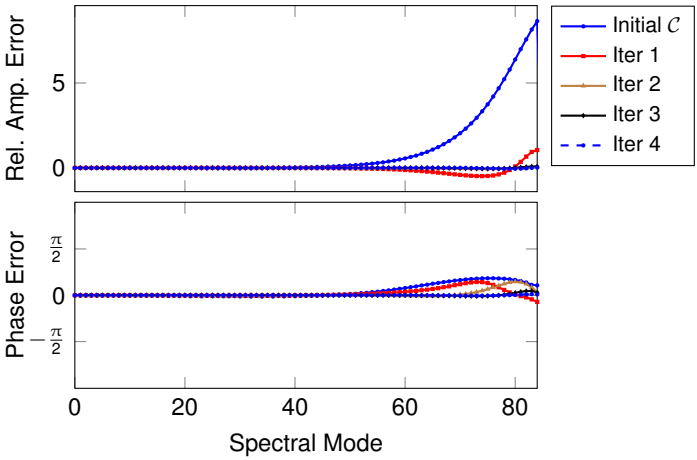
6.6.3 Error Correction Within Parareal

After reviewing the serial errors, it is of interest to investigate how these errors change over the Parareal iterations. This might yield a better insight on the stability of Parareal. For this investigation three specific cases of the the smoothened saw-tooth benchmark are used. These provide examples for the error correction over the Parareal iterations. In all cases the relative amplitude and the phase errors of the propagated waves are plotted over the Fourier modes for the initial guess and the first four Parareal iterations at the time points $t = 0.25$ and $t = 0.75$.

The first case of interest is the second-order SLIRK with $\nu = 1 \times 10^{-4}$. The results are shown in Fig. 6.25. This case is a typical example for all fast



(a) $t = 0.25$



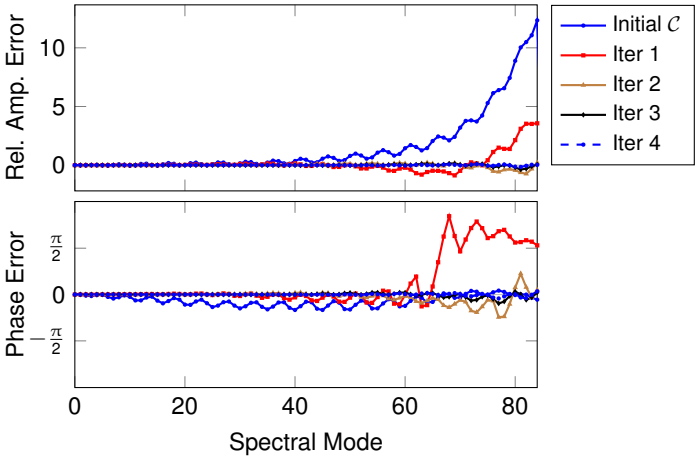
(b) $t = 0.75$

Figure 6.25: Relative amplitude and phase errors of the propagated waves plotted over spectral modes for initial guess and first four Parareal iterations with second-order SLIRK. Shown are two snapshots in time for the viscosity $\nu = 0.0001$

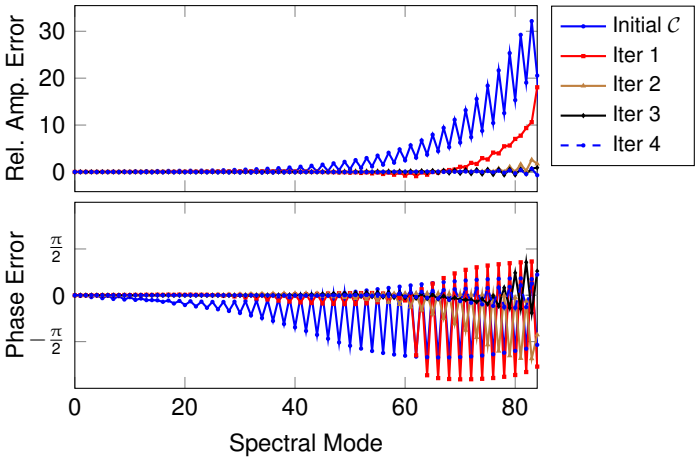
converging Parareal computations. The initial guess shows visible errors in the modes $m > 50$. These errors are nearly the same for both shown time points, which leads to identical error reduction over the Parareal iterations for both time points. The Parareal iterations decrease the relative amplitude error very fast from the initial maximal error of $\varepsilon_{rel} = 8.7$ to the maximal error of $\varepsilon_{rel} = 2.1 \times 10^{-2}$ in four iterations. The fast reduction of the amplitude errors also reduces the error of the missing mass conservation quickly. This is the case for all calculations and with that this error will no longer be discussed. The first iteration decreases the phase error for all modes, however, a large improvement can only be seen for the modes $m > 73$. The acceleration of the modes $m > 80$ is even overcorrected in to a small deceleration. In the following iteration the phase errors of the modes up to $m < 76$ are decreased significantly, where as for the higher modes the error increases again to a acceleration. The next iterations then step by step decrease the phase error. Overall, the phase error is reduced very slowly, as the difference in the maximal phase error of the initial guess and the first two iterations is nearly negligible.

A first case with untypical behavior is the calculation with the first-order SLIRK and $\nu = 1 \times 10^{-4}$. The results in Fig. 6.26 still show an error reduction with consecutive Parareal iterations. However, the phase error increases in the higher modes ($m > 63$) from the initial guess to the first iteration. This overcorrection leading to increased errors as it was already seen with the first case matches the statement of [85], that Parareal converges from above for higher modes.

The phase errors of the initial guess at $t = 0.25$ were already discussed in the previous section. Fig. 6.26a reveals now the line connecting the relative amplitude errors being similar to the one of the phase errors with respect to having the shape expected from the results with the larger viscosity with an additional oscillation on top of it. This oscillation has the same frequency for both errors. Each iteration reduces the relative amplitude errors further. In every iteration the connecting line between the errors is still visibly oscillating. For some spectral modes there is also an over correction such that the amplitude is underpredicted after being overpredicted in the previous iteration. In these cases the absolute value of the error is still reduced. The



(a) $t = 0.25$



(b) $t = 0.75$

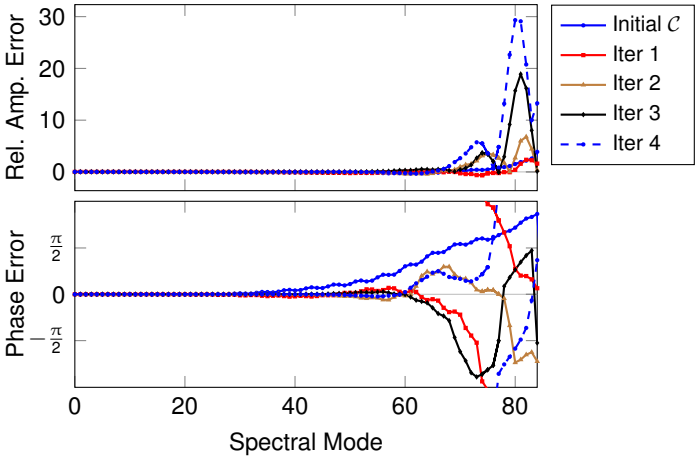
Figure 6.26: Relative amplitude and phase errors of the propagated waves plotted over spectral modes for initial guess and first four Parareal iterations with first-order SLIRK. Shown are two snapshots in time for the viscosity $\nu = 0.0001$

phase errors are not decreased for all spectral modes from the initial guess to the first iteration, as mentioned before. Up to mode $m = 58$ all phase errors are reduced. For all higher modes the errors increase drastically. While the phase error of the initial guess was bound by $|\varepsilon_\Phi| = 0.52$ rad and mostly a deceleration, the error after the first iteration is bound by $|\varepsilon_\Phi| = 2.47$ rad and mostly an acceleration. In the following iterations the error is reduced comparable to the amplitude error. The reduction is fast, but overshoots as well leading to alternating acceleration and deceleration at the same mode. The oscillation is also not dampened strongly over the iterations.

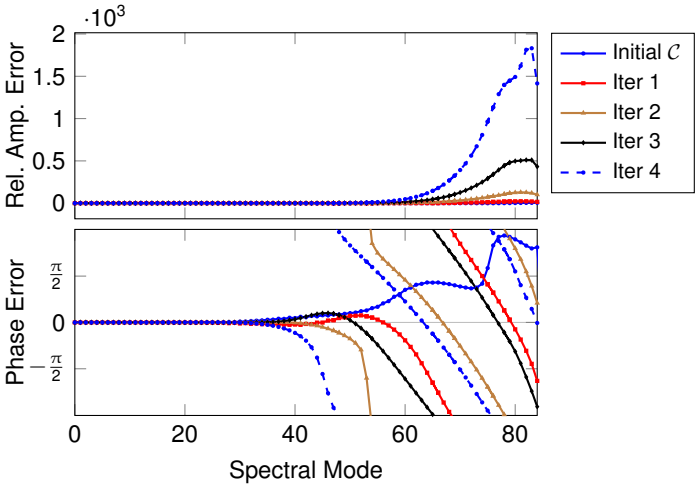
At time point $t = 0.75$ comparable effects of Parareal on the relative amplitude errors can be found, as shown in Fig. 6.26b. The biggest difference qualitatively to the previous figure is the overlaying oscillation being of higher frequency. The connection of the phase errors has an overlaying oscillation also and it has the same frequency, as the one seen with the amplitude errors. Up to mode $m = 63$ the first Parareal iteration improves the error in the phase. For most modes the error reduction is very effective. As seen before, for the modes $m > 63$ the phase error increases during the first iteration. All following iterations then reduce the error again slowly. The phase error reduction appears to sweep from the low to the high modes. As long as there are significant errors in lower modes, the reduction of the higher modes is very small.

Finally, a case with a diverging coarse solver is investigated. This case is the second-order IMEX with $\nu = 1 \times 10^{-4}$. Its errors are provided in Fig. 6.27 for the time points $t = 0.25$ and $t = 0.75$.

The relative amplitude errors at $t = 0.25$ show the initial guess being a stable calculation with reasonable error values ($\varepsilon_{rel} < 3.9$) compared to the ones seen in the previous cases, cf. Fig. 6.27a. Running one iteration of Parareal even reduces the amplitude error for all but one of the high modes. Further iterations lead to increasing errors especially for the modes $m > 68$. The phase errors of the initial guess show the known larger errors for higher spectral modes. In the course of the first Parareal iteration these errors get corrected well for the modes up to $m = 60$. For the higher modes the correction is overshooting massively yielding a deceleration of up to nearly $\varepsilon_\Phi = 2\pi$ rad,



(a) $t = 0.25$



(b) $t = 0.75$

Figure 6.27: Relative amplitude and phase errors of the propagated waves plotted over spectral modes for initial guess and first four Parareal iterations with second-order IMEX. Shown are two snapshots in time for the viscosity $\nu = 0.0001$

from the initial guess error which has a maximal acceleration of $\varepsilon_\Phi = 2.72$ rad. The following iteration reduces the phase errors again for the cost of the increased amplitude errors. After the third and fourth iteration the errors are rising again showing no indication of convergence. With each iteration decelerating errors are changed to accelerating errors and vice versa.

The results at the later time point $t = 0.75$ are comparable, see Fig. 6.27b. Again, the initial guess is a stable calculation with reasonable amplitude errors ($\varepsilon_{rel} < 4.9$) and phase errors ($\varepsilon_\Phi < 2.95$ rad), which are larger for the highest spectral modes. Here, the amplitude error is increased already by the first iteration. It also increases faster over the following iterations reaching a maximum of $\varepsilon_{rel} = 1.83 \times 10^3$ after the fourth iteration. Over the first iteration the phase errors in the modes $m > 56$ are overcorrected from an acceleration to a strong deceleration. In the most extreme case the deceleration is larger than a whole phase. Further iterations increase the phase errors even further, such that after four iterations the large deceleration already starts at mode $m = 38$. Here, the highest mode is placed two full phases behind its analytical position. Comparing these results with Fig. 6.21 shows similarities leading to the conclusion, that the coarse solver reaches its unstable region due to the errors added by the Parareal iterations by overcorrection.

Overall, these three cases show the correction of phase errors in Parareal being not very efficient. It is either relatively slow, as in the first case, or even leads to enlarged errors due to overcorrection as seen with the other two cases. These errors are then again reduced slowly at best.

7 Conclusions and Outlook

The overall goal of this work was to get a better understanding of parallel-in-time methods (PinT) which can be applied to the Navier-Stokes equations. Therefore, the application of parallelization in time techniques to the viscous Burgers equation was investigated, which provides a simpler, but relevant test environment for the Navier-Stokes equations.

The first concept of parallelization in time, which was looked into, is the possibility to exploit degrees of parallelism with the Adomian decomposition method. For this study the Adomian decomposition method (ADM) was applied as an explicit time-stepping scheme. By extracting additional degrees of parallelism within the discretized Adomian decomposition method (DADM) it was possible to reduce its runtime complexity from quadratic to linear.

This reduction makes the DADM a viable competitor to the explicit Runge-Kutta method (ERK). For low-order DADM and ERK schemes the same number of function evaluations has to be computed in serial. In case of high-order schemes, the DADM needs even less function evaluations computed in serial than the ERK. These two methods were examined based on numerical studies revealing both methods having comparable maximal time step sizes. In addition, the error of the DADM was shown to be slightly larger than that of the EKR. This difference increases with an increasing order of accuracy of the schemes. However, these larger errors can be circumvented by increasing the order p of the DADM scheme by one to $p + 1$. By exploiting the found parallelism, the DADM of one order higher than the ERK still needs less function evaluations computed in serial. Increasing the order of the DADM scheme is easy to accomplish, as it is a straightforward procedure. These results show, for simulations where a high-order of accuracy is desired, the possibility for the DADM being used to reduce the time-to-solution.

In [35] the possibility of increasing the maximal time step width of the ADM using Padé's rational approximation is mentioned. Combining the approximation with the exploitation of the additional degrees of parallelism might increase the viability of the DADM as an explicit time-stepping scheme further and decrease the time-to-solution for high-order schemes even more.

The second approach of parallelization in time investigated here is the potential of the semi-Lagrangian implicit Runge-Kutta method as the coarse solver for the Parareal algorithm. The focus of this study lay on the benefits of the semi-Lagrangian formulation and its stability as a coarse solver for advection dominated problems. Therefore, three benchmarks with different characteristics were used in the region of small viscosities whereof two are based on manufactured solutions.

The semi-Lagrangian formulation combined with an implicit Runge-Kutta method (SLIRK) was compared to an explicit (ERK) and an implicit-explicit (IMEX) Runge-Kutta method as a coarse solver. These coarse solvers were tested with first and second-order of accuracy. For all computations the IMEX was chosen as the fine solver. With the investigated test cases the ERK method was not stable as a coarse solver and, thus, did not lead to any speedup regarding the number of iterations to convergence. The SLIRK method being the favorite over the IMEX method was observed for the benchmark runs for cases where the IMEX method was unstable as a coarse solver at least in some of the Parareal iterations. In these cases the SLIRK method converged with far less iterations due to its stable behavior. However, a similar number of iterations to convergence is found for SLIRK and IMEX in cases where both coarse solvers were stable for all Parareal iterations. The difference between the first and second-order schemes in these cases is so small, that the additional effort, especially for the second-order SLIRK, which is computationally more expensive and needs additional communication, is not justified. The choice between the first-order SLIRK and the first-order IMEX has then to be made based on their respective computational cost.

Compared to the IMEX method a larger range of viscosities is made suitable to the Parareal algorithm by the stability of the SLIRK method. However, the instability of Parareal itself still leads to an increasing number of iter-

ations to convergence for a decreasing viscosity. With the inviscid Burgers equation ($\nu = 0$) even the stable SLIRK method needs approximately half of the maximal number of iterations to converge. Hence, no large speedup can be expected in the fully advective case.

The computations with the inviscid Burgers equation showed the efficiency potential of Parareal decreasing if the high Fourier modes are important for the solution. Based on this, the wave propagation characteristics of the coarse solvers and Parareal were investigated. Here, the errors in the prediction of the wave position, especially for the high Fourier modes, leading to slow convergence of Parareal was shown. These errors in the phase are corrected very slowly by Parareal, where as errors in the amplitude are corrected fast.

In this work only the speedup potential based on the number of iterations to convergence is discussed, since the implementation of the Parareal algorithm used is run serially. In further work the actual speedup has to be investigated considering the computational cost of the SLIRK method and the necessary communication.

This speedup test then could be combined with a comparison to fully implicit solvers. These have also the stability property of the SLIRK method and, thus, should be beneficial to the iterations to convergence of the Parareal algorithm as well. However, the open question is, whether the implicit solvers are computationally less expensive than the SLIRK method.

Building on the presented work, the transferability of the positive effect of the SLIRK method compared to the IMEX method to other parallel-in-time methods can be investigated. A positive effect is expected especially for closely related PinT methods. A first step would be examining the possible speedup with the non-intrusive MGRIT method [45].

Finally, a logical next step would be to apply the Parareal algorithm with the SLIRK coarse solver to the incompressible Navier-Stokes equations. Following the results of this work, speedup potential should be found up to high Reynolds numbers. The potential is expected to decrease with an increasing Reynolds number, but it has to be investigated up to which number a realistic potential can be found.

Bibliography

[1] Abbaoui, K.; Cherruault, Y.: Convergence of Adomian's method applied to nonlinear equations, *Mathematical and Computer Modelling*, 20(9):69–73, 1994.

[2] Abbasbandy, S.; Darvishi, M.T.: A numerical solution of Burgers' equation by modified Adomian method, *Applied Mathematics and Computation*, 163(3):1265–1272, 2005.

[3] Ablowitz, M.J.: *Nonlinear Dispersive Waves: Asymptotic Analysis and Solutions*, Cambridge Texts in Applied Mathematics, Cambridge University Press, Cambridge, UK, 2011.

[4] Adomian, G.: A new approach to nonlinear partial differential equations, *Journal of Mathematical Analysis and Applications*, 102(2):420–434, 1984.

[5] Adomian, G.: *Nonlinear Stochastic Operator Equations*, Academic Press, New York, 1986.

[6] Adomian, G.: Solution of physical problems by decomposition, *Computers & Mathematics with Applications*, 27(9):145–154, 1994.

[7] Adomian, G.: *Solving Frontier Problems of Physics: The Decomposition Method*, Springer Science+Business Media, Dordrecht, 1994.

[8] Adomian, G.: Explicit solutions of nonlinear partial differential equations, *Applied Mathematics and Computation*, 88(2):117–126, 1997.

[9] Akpan, I.: Adomian decomposition approach to the solution of the Burger's equation, *American Journal of Computational Mathematics*, 5:329–335, 2015.

[10] Ascher, U.M.; Ruuth, S.J.; Spiteri, R.J.: Implicit-explicit Runge-Kutta methods for time-dependent partial differential equations, *Applied Numerical Mathematics*, 25(2-3):151–167, 1997.

[11] Baffico, L.; Bernard, S.; Maday, Y.; Turinici, G.; Zérah, G.: Parallel-in-time molecular-dynamics simulations, *Physical Review E*, 66:057701, 2002.

[12] Bal, G.: On the convergence and the stability of the Parareal algorithm to solve partial differential equations, in R. Kornhuber; R.W. Hoppe; J. Périaux; O. Pironneau; O. Widlund; J. Xu (Eds.), *Domain Decomposition Methods in Science and Engineering*, vol. 40 of *Lecture Notes in Computational Science and Engineering*, pp. 426–432, Springer, Berlin, 2005.

[13] Bal, G.; Maday, Y.: A "Parareal" time discretization for non-linear PDE's with application to the pricing of an american put, in L.F. Pavarino; A. Toselli (Eds.), *Recent Developments in Domain Decomposition Methods*, pp. 189–202, Springer Berlin Heidelberg, Berlin, Heidelberg, 2002.

[14] Barros, S.R.; Peixoto, P.S.: Computational aspects of harmonic wavelet Galerkin methods and an application to a precipitation front propagation model, *Computers & Mathematics with Applications*, 61(4):1217–1227, 2011.

[15] Batchelor, G.K.: *An Introduction to Fluid Dynamics*, Cambridge Mathematical Library, Cambridge University Press, Cambridge, 1973.

[16] Bateman, H.: Some recent researches on the motion of fluids, *Monthly Weather Review*, 43(4):163–170, 1915.

[17] Bauer, P.; Thorpe, A.; Brunet, G.: The quiet revolution of numerical weather prediction, *Nature*, 525(7567):47–55, 2015.

[18] Bedez, M.; Belhachmi, Z.; Haeberlé, O.; Greget, R.; Moussaoui, S.; Bouteiller, J.M.; Bischoff, S.: A fully parallel in time and space algorithm for simulating the electrical activity of a neural tissue, *Journal of Neuroscience Methods*, 257:17–25, 2016.

[19] Bonaventura, L.: An introduction to semi-Lagrangian methods for geophysical scale flows, *Lecture Notes, ERCOFTAC Leonhard Euler Lectures, SAM-ETH Zurich*, 2004.

[20] Briggs, W.L.; Henson, V.E.; McCormick, S.F.: *A multigrid tutorial*, Philadelphia, PA: SIAM, 2nd edn., 2000.

[21] Burgers, J.M.: A mathematical model illustrating the theory of turbulence, vol. 1 of *Advances in Applied Mechanics*, pp. 171–199, Elsevier, 1948.

[22] Burgers, J.M.: *The Nonlinear Diffusion Equation: Asymptotic Solutions and Statistical Problems*, Springer, Netherlands, 1974.

[23] Butcher, J.C.: Implicit Runge-Kutta processes, *Mathematics of Computation*, 18(85):50–64, 1964.

[24] Butcher, J.C.: *Numerical Method for Ordinary Differential Equations*, John Wiley & Sons, Ltd, 2nd edn., 2008.

[25] Chen, F.; Hesthaven, J.S.; Zhu, X.: On the use of reduced basis methods to accelerate and stabilize the Parareal method, in A. Quarteroni; G. Rozza (Eds.), *Reduced Order Methods for Modeling and Computational Reduction*, vol. 9 of *MS&A - Modeling, Simulation and Applications*, pp. 187–214, Springer International Publishing, 2014.

[26] Cheng, M.; Scott, K.; Sun, Y.; Wu, B.: Explicit solution of nonlinear electrochemical models by the decomposition method, *Chemical Engineering & Technology*, 25(12):1155–1160, 2002.

[27] Cherruault, Y.; Adomian, G.: Decomposition methods: A new proof of convergence, *Mathematical and Computer Modelling*, 18(12):103–106, 1993.

[28] Christlieb, A.J.; Macdonald, C.B.; Ong, B.W.: Parallel high-order integrators, *SIAM Journal on Scientific Computing*, 32(2):818–835, 2010.

[29] Cole, J.D.: On a quasi-linear parabolic equation occuring in aerodynamics, *Quarterly of Applied Mathematics*, 9:225–236, 1951.

[30] Cortial, J.; Farhat, C.: A time-parallel implicit method for accelerating the solution of non-linear structural dynamics problems, *International Journal for Numerical Methods in Engineering*, 77(4):451–470, 2008.

[31] Côté, J.: Time-parallel algorithms for weather prediction and climate simulation, `https://www.newton.ac.uk/files/seminar/20121023121012359-153396.pdf`, 2012, Accessed: 03 May 2017.

[32] Courant, R.; Friedrichs, K.; Lewy, H.: Über die partiellen Differenzengleichungen der mathematischen Physik, *Mathematische Annalen*, 100:32–74, 1928.

[33] Dahmen, W.; Reusken, A.: *Numerik für Ingenieure und Naturwissenschaftler*, Springer Berlin Heidelberg New York, 2006.

[34] Dai, X.; Maday, Y.: Stable Parareal in time method for first- and second-order hyperbolic systems, *SIAM Journal on Scientific Computing*, 35(1):A52–A78, 2013.

[35] de Sousa Basto, M.J.F.: *Adomian Decomposition Method, Nonlinear Equations and Spectral Solutions of Burgers Equation*, Ph.D. thesis, Faculdade de Engenharia da Universidade do Porto, 2006.

[36] Dolean, V.; Jolivet, P.; Nataf, F.: *An Introduction to Domain Decomposition Methods: Algorithms, Theory, and Paralell Implementation*, SIAM, 2015.

[37] Durran, D.R.: *Numerical Methods for Fluid Dynamics: With Applications to Geophysics*, Texts in Applied Mathematics, Springer New York, 2010.

[38] Dutt, A.; Greengard, L.; Rokhlin, V.: Spectral deferred correction methods for ordinary differential equations, *BIT Numerical Mathematics*, 40(2):241–266, 2000.

[39] El-Sayed, S.M.; Kaya, D.: On the numerical solution of the system of two-dimensional Burgers' equations by the decomposition method, *Applied Mathematics and Computation*, 158(1):101–109, 2004.

[40] El-Tawil, M.A.; Bahnasawi, A.A.; Abdel-Naby, A.: Solving Riccati differential equation using Adomian's decomposition method, *Applied Mathematics and Computation*, 157(2):503–514, 2004.

[41] Emmett, M.; Minion, M.L.: Toward an efficient parallel in time method for partial differential equations, *Communications in Applied Mathematics and Computational Science*, 7:105–132, 2012.

[42] Farhat, C.; Chandesris, M.: Time-decomposed parallel time-integrators: theory and feasibility studies for fluid, structure, and fluid-structure applications, *International Journal for Numerical Methods in Engineering*, 58(9):1397–1434, 2003.

[43] Farhat, C.; Cortial, J.; Dastillung, C.; Bavestrello, H.: Time-parallel implicit integrators for the near-real-time prediction of linear structural dynamic responses, *International Journal for Numerical Methods in Engineering*, 67:697–724, 2006.

[44] Fischer, P.F.; Hecht, F.; Maday, Y.: A parareal in time semi-implicit approximation of the Navier-Stokes equations, in R. Kornhuber; R.W. Hoppe; J. Périaux; O. Pironneau; O. Widlund; J. Xu (Eds.), *Domain Decomposition Methods in Science and Engineering*, vol. 40 of *Lecture Notes in Computational Science and Engineering*, pp. 433–440, Springer, Berlin, 2005.

[45] Friedhoff, S.; Falgout, R.D.; Kolev, T.V.; MacLachlan, S.P.; Schroder, J.B.: A multigrid-in-time algorithm for solving evolution equations in parallel, in *Presented at: Sixteenth Copper Mountain Conference on Multigrid Methods, Copper Mountain, CO, United States, Mar 17 - Mar 22, 2013*, 2013.

[46] Gander, M.J.: Analysis of the Parareal algorithm applied to hyperbolic problems using characteristics, *Boletin de la Sociedad Espanola de Matemática Aplicada*, 42:21–35, 2008.

[47] Gander, M.J.: 50 years of time parallel time integration, in *Multiple Shooting and Time Domain Decomposition*, Springer, 2015.

[48] Gander, M.J.; Güttel, S.: PARAEXP: A parallel integrator for linear initial-value problems, *SIAM Journal on Scientific Computing*, 35(2):C123–C142, 2013.

[49] Gander, M.J.; Güttel, S.; Petcu, M.: A nonlinear ParaExp algorithm, MIMS EPrint: 2017.17 `http://eprints.ma.man.ac.uk/2550/01/covered/MIMS_ep2017_17.pdf`, 2017.

[50] Gander, M.J.; Hairer, E.: Nonlinear Convergence Analysis for the Parareal Algorithm, in U. Langer; O. Widlund; D. Keyes (Eds.), *Domain Decomposition Methods in Science and Engineering*, vol. 60 of *Lecture Notes in Computational Science and Engineering*, pp. 45–56, Springer, 2008.

[51] Gander, M.J.; Kulchytska-Ruchka, I.; Niyonzima, I.; Schöps, S.: A new Parareal algorithm for problems with discontinuous sources, arXiv:1803.05503 [math.NA], 2018.

[52] Gander, M.J.; Petcu, M.: Analysis of a Krylov subspace enhanced Parareal algorithm for linear problems, *ESAIM Proceedings*, 25:114–129, 2008.

[53] Gander, M.J.; Vandewalle, S.: Analysis of the Parareal time-parallel time-integration method, *SIAM Journal on Scientific Computing*, 29(2):556–578, 2007.

[54] Gottlieb, D.; Orszag, S.A.: *Numerical Analysis of Spectral Methods: Theory and Applications*, CBMS-NSF Regional Conference Series in Applied Mathematics, Society for Industrial and Applied Mathematics, 1977.

[55] Guellal, S.; Grimalt, P.; Cherruault, Y.: Numerical study of Lorenz's equation by the Adomian method, *Computers & Mathematics with Applications*, 33(3):25–29, 1997.

[56] Hairer, E.; Wanner, G.: *Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems*, Springer, Berlin, Heidelberg, 1996.

[57] Hopf, E.: The partial differential equation $u_t + uu_x = \mu_{xx}$, *Communications on Pure and Applied Mathematics*, 3(3):201–230, 1950.

[58] Hortal, M.: The development and testing of a new two-time-level semi-Lagrangian scheme (SETTLS) in the ECMWF forecast model, *Quarterly Journal of the Royal Meteorological Society*, 128(583):1671–1687, 2002.

[59] Horton, G.; Vandewalle, S.: A space-time multigrid method for parabolic partial differential equations, *SIAM Journal on Scientific Computing*, 16(4):848–864, 1995.

[60] Horton, G.; Vandewalle, S.; Worley, P.: An algorithm with polylog parallel complexity for solving parabolic partial differential equations, *SIAM Journal on Scientific Computing*, 16(3):531–541, 1995.

[61] Jiao, Y.; Yamamoto, Y.; Dang, C.; Hao, Y.: An aftertreatment technique for improving the accuracy of Adomian's decomposition method, *Computers & Mathematics with Applications*, 43(6):783–798, 2002.

[62] Kaya, D.; Yokus, A.: A numerical comparison of partial solutions in the decomposition method for linear and nonlinear partial differential equations, *Mathematics and Computers in Simulation*, 60(6):507–512, 2002.

[63] Kaya, D.; Yokus, A.: A decomposition method for finding solitary and periodic solutions for a coupled higher-dimensional Burgers equations, *Applied Mathematics and Computation*, 164(3):857–864, 2005.

[64] Kennedy, C.A.; Carpenter, M.H.: Additive Runge-Kutta schemes for convection-diffusion-reaction equations, *Applied Numerical Mathematics*, 44(1):139–181, 2003.

[65] Kooij, G.L.; Botchev, M.A.; Geurts, B.J.: A block Krylov subspace implementation of the time-parallel Paraexp method and its extension for nonlinear partial differential equations, *Journal of Computational and Applied Mathematics*, 316:229–246, 2017, Selected Papers from NUMDIFF-14.

[66] Kooij, G.L.; Botchev, M.A.; Geurts, B.J.: An exponential time integrator for the incompressible Navier–Stokes equation, *SIAM Journal on Scientific Computing*, 40(3):B684–B705, 2018.

[67] Lamb, H.: *Hydrodynamics*, Cambridge University Press, Cambridge, 6th edn., 1994.

[68] Leonard, B.P.; Lock, A.P.; MacVean, M.K.: Conservative explicit unrestricted-time-step multidimensional constancy-preserving advection schemes, *Monthly Weather Review*, 124(11):2588–2606, 1996.

[69] LeVeque, R.J.: *Finite Difference Methods for Ordinary and Partial Differential Equations: Steady-State and Time-Dependent Problems*, SIAM e-books, Society for Industrial and Applied Mathematics, Philadelphia, USA, 2007.

[70] Lin, S.J.; Rood, R.B.: Multidimensional flux-form semi-Lagrangian transport schemes, *Monthly Weather Review*, 124(9):2046–2070, 1996.

[71] Lions, J.L.; Maday, Y.; Turinici, G.: A "parareal" in time discretization of PDE's, *Comptes Rendus de l'Académie des Sciences - Series I - Mathematics*, 332:661–668, 2001.

[72] Lunet, T.; Bodart, J.; Gratton, S.; Vasseur, X.: Time-parallel simulation of the decay of homogeneous turbulence using Parareal with spatial coarsening, *Computing and Visualization in Science*, 19(1):31–44, 2018.

[73] Maday, Y.; Turinici, G.: Parallel in time algorithms for quantum control: Parareal time discretization scheme, *International Journal of Quantum Chemistry*, 93(3):223–228, 2003.

[74] Nair, R.D.; Scroggs, J.S.; Semazzi, F.H.M.: Efficient conservative global transport schemes for climate and atmospheric chemistry models, *Monthly Weather Review*, 130(8):2059–2073, 2002.

[75] Nievergelt, J.: Parallel methods for integrating ordinary differential equations, *Communications of the ACM*, 7(12):731–733, 1964.

[76] Norman, M.R.; Nair, R.D.: Inherently conservative nonpolynomial-based remapping schemes: Application to semi-Lagrangian transport, *Monthly Weather Review*, 136(12):5044–5061, 2008.

[77] Peixoto, P.S.; Barros, S.R.M.: On vector field reconstructions for semi-Lagrangian transport methods on geodesic staggered grids, *Journal of Computational Physics*, 273:185–211, 2014.

[78] Press, W.H.; Flannery, B.P.; Teukolsky, S.A.; Vetterling, W.T.; et al.: *Numerical Recipes*, vol. 3, Cambridge University Press, Cambridge, 1989.

[79] Rančić, M.: Semi-Lagrangian piecewise biparabolic scheme for two-dimensional horizontal advection of a passive scalar, *Monthly Weather Review*, 120:1394–1406, 1992.

[80] Reynolds-Barredo, J.M.; Newman, D.E.; Sanchez, R.; Jenko, F.: A novel, semilagrangian, coarse solver for the Parareal technique and its application to 2D fluid drift-wave (BETA) and 5d gyrokinetic (GENE), turbulence simulations, in *APS Meeting Abstracts*, vol. 1, p. 8128P, 2013.

[81] Richardson, L.F.: The approximate arithmetical solution by finite differences of physical problems involving differential equations, with an application to the stresses in a masonry dam, *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 210(459-470):307–357, 1911.

[82] Richardson, L.F.; Gaunt, J.A.: The deferred approach to the limit, *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 226(636-646):299–361, 1927.

[83] Roache, P.: Code verification by the method of manufactured solutions, *Journal of Fluids Engineering*, 124(1):4–10, 2002.

[84] Rozhdestvenskii, B.L.: Hyperbolic partial differential equation, Encyclopedia of Mathematics, `http://www.encyclopediaofmath.org/index.php?title=Hyperbolic_partial_differential_equation&oldid=28218`, Accessed: 31.01.2018.

[85] Ruprecht, D.: Wave propagation characteristics of Parareal, *Computing and Visualization in Science*, 19(1):1–17, 2018.

[86] Ruprecht, D.; Krause, R.: Explicit parallel-in-time integration of a linear acoustic-advection system, *Computers & Fluids*, 59(0):72–83, 2012.

[87] Salari, K.; Knupp, P.: Code verification by the method of manufactured solutions, *Tech. Rep. SAND2000-1444*, Sandia National Labs., Albuquerque, NM, United States, 2000.

[88] Schäfer, M.: *Computational Engineering: Introduction to Numerical Methods*, Springer, Berlin Heidelberg, 2006.

[89] Schmitt, A.; Schreiber, M.; Peixoto, P.; Schäfer, M.: A numerical study of a semi-Lagrangian Parareal method applied to the viscous Burgers equation, *Computing and Visualization in Science*, 19(1):45–57, 2018.

[90] Schmitt, A.; Schreiber, M.; Schäfer, M.: Additional degrees of parallelism within the Adomian decomposition method, in M. Schäfer; M. Behr; M. Mehl; B. Wohlmut (Eds.), *Recent Advances in Computational Engineering*, vol. 124 of *Lecture Notes in Computational Science and Engineering*, pp. 111–126, Springer International Publishing, Berlin, 2018.

[91] Schöps, S.; Niyonzima, I.; Clemens, M.: Parallel-in-time simulation of eddy current problems using parareal, arXiv:1706.05750v1 [cs.CE], 2017.

[92] Schreiber, M.; Peixoto, P.; Schmitt, A.: SWEET webpage, `https://schreiberx.github.io/sweetsite/index.html`, Accessed: 15.01.2018.

[93] Schreiber, M.; Peixoto, P.S.; Haut, T.; Wingate, B.: Beyond spatial scalability limitations with a massively parallel method for linear oscillatory problems, *The International Journal of High Performance Computing Applications*, 2016.

[94] Shawagfeh, N.; Kaya, D.: Comparing numerical methods for the solutions of systems of ordinary differential equations, *Applied Mathematics Letters*, 17(3):323–328, 2004.

[95] Speck, R.; Ruprecht, D.; Emmett, M.; Bolten, M.; Krause, R.: A space-time parallel solver for the three-dimensional heat equation, in *Parallel Computing: Accelerating Computational Science and Engineering (CSE)*, vol. 25 of *Advances in Parallel Computing*, pp. 263–272, IOS Press, 2014.

[96] Staff, G.A.; Rønquist, E.M.: Stability of the Parareal algorithm, in R. Kornhuber; R.W. Hoppe; J. Périaux; O. Pironneau; O. Widlund; J. Xu (Eds.), *Domain Decomposition Methods in Science and Engineering*, vol. 40 of *Lecture Notes in Computational Science and Engineering*, pp. 449–456, Springer, Berlin, 2005.

[97] Staniforth, A.; Côté, J.: Semi-Lagrangian integration schemes for atmospheric models - A review, *Monthly weather review*, 119(9):2206–2223, 1991.

[98] Steiner, J.; Ruprecht, D.; Speck, R.; Krause, R.: Convergence of Parareal for the Navier-Stokes equations depending on the Reynolds number, in A. Abdulle; S. Deparis; D. Kressner; F. Nobile; M. Picasso (Eds.), *Numerical Mathematics and Advanced Applications - ENUMATH 2013*, vol. 103 of *Lecture Notes in Computational Science and Engineering*, pp. 195–202, Springer, Cham, 2015.

[99] Strang, G.: On the construction and comparison of difference schemes, *SIAM Journal on Numerical Analysis*, 5(3):506–517, 1968.

[100] Sutter, H.: The free lunch is over: A fundamental turn toward concurrency in software, *Dr. Dobb's journal*, 30(3):202–210, 2005.

[101] Swarztrauber, P.N.; Sweet, R.A.: The Fourier and cyclic reduction methods for solving Poisson's equation, in *Handbook of Fluid Dynamics and Fluid Machinery*, John Wiley & Sons, New York, NY, USA, 1996.

[102] Trindade, J.M.F.; Pereira, J.C.F.: Parallel-in-time simulation of the unsteady Navier-Stokes equations for incompressible flow, *International Journal for Numerical Methods in Fluids*, 45(10):1123–1136, 2004.

[103] Trindade, J.M.F.; Pereira, J.C.F.: Parallel-in-time simulation of two-dimensional, unsteady, incompressible laminar flows, *Numerical Heat Transfer, Part B: Fundamentals*, 50(1):25–40, 2006.

[104] Vadasz, P.; Olek, S.: Convergence and accuracy of Adomian's decomposition method for the solution of Lorenz equations, *International Journal of Heat and Mass Transfer*, 43(10):1715–1734, 2000.

[105] Vandewalle, S.; Horton, G.: Fourier mode analysis of the multigrid waveform relaxation and time-parallel multigrid methods, *Computing*, 54(4):317–330, 1995.

[106] Wedi, N.P.; Bauer, P.; Deconinck, W.; Diamantakis, M.; Hamrud, M.; Kuehnlein, C.; Malardel, S.; Mogensen, K.; Mozdzynski, G.; Smolarkiewicz, P.K.: The modelling infrastructure of the integrated forecasting system: Recent advances and future challenges, *Technical Memorandum 760*, European Centre for Medium-Range Weather Forecasts, Reading, UK, 2015.

[107] Wesseling, P.: *Principles of Computational Fluid Dynamics*, Springer Series in Computational Mathematics, Springer Berlin Heidelberg, 2009.

[108] Williamson, D.L.: The evolution of dynamical cores for global atmospheric models, *Journal of the Meteorological Society of Japan. Ser. II*, 85:241–269, 2007.

[109] Wood, W.L.: An exact solution for Burger's equation, *Communications in Numerical Methods in Engineering*, 22(7):797–798, 2006.

[110] Zerroukat, M.; Wood, N.; Staniforth, A.: SLICE: A Semi-Lagrangian Inherently Conserving and Efficient scheme for transport problems, *Quarterly Journal of the Royal Meterological Society*, 128(586):2801–2820, 2002.

[111] Zhu, H.; Shu, H.; Ding, M.: Numerical solutions of two-dimensional Burgers' equations by discrete Adomian decomposition method, *Computers & Mathematics with Applications*, 60(3):840–848, 2010.

List of Figures

List of Tables